Money-Based Coordination of Network Packets*

Pavlos S. Efraimidis and Remous-Aris Koutsiamanis

Dept. of Electrical and Computer Engineering
Democritus Univ. of Thrace, 67100 Xanthi, Greece
{pefraimi,akoutsia}@ee.duth.gr

Abstract. In this work, we apply a common economic tool, namely money, to coordinate network packets. In particular, we present a network economy, called PacketEconomy, where each flow is modeled as a population of rational network packets, and these packets can self-regulate their access to network resources by mutually trading their positions in router queues. We consider a corresponding Markov model of trade and show that there are Nash equilibria (NE) where queue positions and money are exchanged directly between the network packets. This simple approach, interestingly, delivers significant improvements for packets and routers.

1 Introduction

It is known that a large number of independent flows is constantly competing on the Internet for network resources. Without any central authority to regulate its operation, the available network resources of the Internet are allocated by independent routers to the flows in a decentralized manner. Internet flows may submit at any time an arbitrary amount of packets to the network and then adjust their packet rate with an appropriate flow control algorithm, like the AIMD-based algorithms for TCP-flows. The apparent lack of coordination between the independent flows leads the Internet to an "anarchic" way of operation and gives rise to issues and problems that can be addressed with concepts and tools from algorithmic game theory.

Two representative works on applying game theory to network problems are [15,19]. Certain game-theoretic approaches to congestion problems of the Internet, and especially the TCP/IP protocol suite, are discussed in [20,1,8,6]. A combinatorial perspective on Internet congestion problems is given in [12]. The focus of the above works and the present paper is on sharing the network resources between selfish flows. In this work, however, we propose an economy where packets belonging to selfish flows may interact directly with each other.

* The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement no 264226. [Project Title: Space Internetworking Center-SPICE]. This paper reflects only the views of the authors. The Union is not liable for any use that may be made of the information contained.

P.G. Spirakis and M. Serna (Eds.): CIAC 2013, LNCS 7878, pp. 197–208, 2013.

The use of economic tools like pricing, tolls and taxes as a means to regulate the operation of networks and/or to support quality of service (QoS) functionalities in the presence of selfish flows is, for example, discussed in [18,9,4,3,16,17]. In particular, the Paris Metro Pricing approach - using pricing to manage traffic in the Paris Metro - is adapted to computer networks in [18]. A smart market for buying priority in congested routers is presented in [16]. In [4,3] taxes are used to influence the behavior of selfish flows in a different network model. An important issue identified in [3] is that taxes may cause disutility to network users unless the collected taxes can be feasibly returned to the users. In our economic model this issue is naturally solved; trades take place between the flows, so the money is always in the possession of the flows.

In this work, we apply a common economic tool, namely money, to coordinate network packets. This is in contrast to much of the existing literature, which aims to impose charges on Internet traffic, and to our knowledge, this is the first work to propose economic exchanges directly between packets. In particular, we present a network economy, called PacketEconomy, where ordinary network packets can trade their positions in router queues. The role of money in this approach is to facilitate the trades between the network packets. Queue positions and money are exchanged directly between the packets while the routers simply carry out the trades. We show that, in this economy, packets can self-regulate their access to network resources and obtain better services at equilibrium points.

In their seminal work, Kiyotaki and Wright [13] examine the emergence of money as a medium of exchange in barter economies. Subsequently, Gintis [10,11] generalizes the Kiyotaki-Wright model by combining Markov chain theory and game theory. Inspired by the above works, we propose the PacketEconomy where money is used as a coordination mechanism for network packets and prove that there are Nash equilibria where trades are performed to the benefit of all the flows. In the PacketEconomy, specialization - the reason for the emergence of money as per Adam Smith ([21, Chapter 4], cited in [13]) - originates from the diverse QoS requirements of network flows. In particular, the various types of PacketEconomy flows differ in their tolerance for packet delays.

Contribution. The main contributions of this work are:

- A new game-theoretic model representing network packets as populations of rational agents. In this model, a network flow is represented as a population of in-flight packets that can make bilateral trades with other packets.
- Application of bilateral trades and virtual money at a microeconomic level to support better coordination of rational network packets.
- Application of an interesting combination of ergodic Markov chains and strategic games within the context of network games.

Outline. We describe in Section 2 the PacketEconomy and analyze in Section 3 a representative scenario of it. The effect of trades is discussed in Section 4. Concluding remarks are given in Section 5. Due to lack of space, some proofs have been omitted and can be found in the long version of this work [5].

2 An Economy for Packets

The PacketEconomy is comprised of a network model with selfish flows, a queue that supports packet trades, a currency and a specific economic goal. The solution concept is the Nash equilibrium (NE), i.e., a profile of the game in which no player has anything to gain by changing only his/her own strategy unilaterally.

The Network Model. We assume a one-hop network with a router R and a set of N flows, as shown in Figure 1. This setting is equivalent to the common dumbbell topology used for the analysis of many network scenarios, including the seminal paper of Chiu and Jain [2] on the AIMD algorithm. The router R has a FIFO DropTail queue with a maximum capacity of q packets and operates in rounds. In each round, the first packet (the packet at position 0 of the queue) is served. At the end of the round, the first packet reaches its destination. Packets that arrive at the router are added to the end of the queue.

Packet Trades. At the beginning of each round all packets in the queue are shifted one position ahead. A packet that enters the queue in this round, occupies the first free (after the shift) position at the end of the queue. After the shift, the packet that has reached position zero is served, while the other packets in the router queue are simply waiting. These idle packets can engage in trades. During each router round a fixed number b of trading periods take place. In each trading period the idle packets are matched randomly in pairs with a predefined pairing scheme. Each packet pair can perform a trade, as shown in Figure 2, provided the negotiation performed between them leads to an agreement. The way the trades take place at a microeconomic level between paired packets resembles the models of [10,13] where agents meet in random pairs and can make trades.

Packet Delay. The delay d_p of a packet p that starts at position k of the zero-based queue and does not make any trade is $k + 1$ rounds (Figure 3a). If, however, the packet engages in trades and buys a total of r_b router rounds and sells r_s router rounds, then its delay d_p, including the time to be served, becomes $d_p = k + 1 + r_s - r_b$ rounds. A packet may have an upper bound $d_{p,\max}$ on its

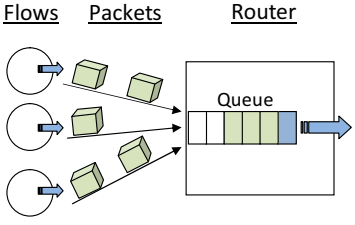


Fig. 1. The network model with the flows, their packets, the router, and the queue

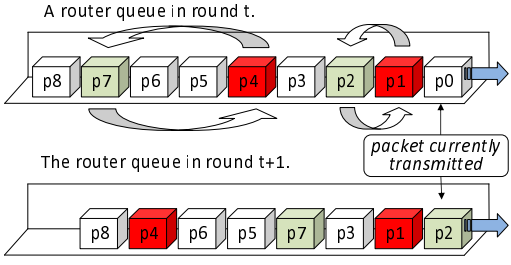


Fig. 2. The state of a router queue in two successive rounds. In round t, two trades take place; one between the packet pair (p1,p2) and one between the pair (p4,p7).

delay; for delays larger than $d_{p,\max}$ the value of the packet becomes zero and the packet will not voluntarily accept such delays (that is, it will not sell).

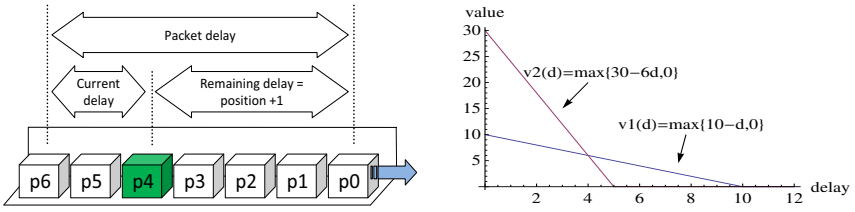
Details. The router operates in rounds and can serve one packet in each one. All packets are assumed to be of the same size and no queue overflows occur. In generating the random packet pairs, the use of predefined pairing reduces the computational burden and avoids stable marriage problems. We make the plausible assumption that flows with different QoS preferences are competing for the network resources. We also make the assumption that the preferences of each flow can be expressed with a utility function for its packets. Thus, packets with different utility functions will, in general, co-exist in the router queue.

Packet Values. For each packet p there is a flow-specific decreasing function $v_p(d)$ which determines the value of p, given its delay d. The value function of each flow must be encoded onto each packet. Thus, its computational requirements should be low in order not to overload the router. A class of simple value functions are $v_p(d) = \max\{v_{\max} - c_p \cdot d, 0\}$ where c_p is the cost per unit of delay (Figure 3b). The value of a packet can be calculated anytime during the packet's journey via the $v_p(d)$ function.

In the PacketEconomy every packet has its compensatory price p. For prices lower than p, the packet is ready to buy better queue positions and for prices higher than p it is ready to sell its position, provided that the extra delay will not cause it to exceed its maximum delay limit.

Inventories. Every time a packet is delivered in time, wealth is created for the flow that owns the packet. Each packet p has an inventory $I_p(t)$ containing two types of indivisible goods or resources; the packet delay $d_p(t)$ and the money account $a_p(t)$. Note that delay bears negative value, whereas money represents positive value. We assume positive integer constants s_a, s_b and s_d, such that $a_p(t) \in \{-s_a, \ldots, s_b\}$ and $d_p(t) \in \{0, \ldots, s_d\}$. The inventory also contains the current position $pos_p(t)$ of the packet in the queue if it is waiting in the queue. When the packet reaches its destination, the contents of the inventory of the packet are used to determine its utility. This utility is then reimbursed to the flow that owns the packet and a new packet of the same flow enters the queue.

Benefit and Utility. Every packet has two types of resources that bear value, the packet value and the budget of a packet. We define the notion of the packet



(a) Packet delay terminology for p4 (b) Two simple packet value functions

Fig. 3. Delays and Packet Values

benefit as the sum of the value of a packet plus/minus its budget. Then we use the benefit concept to define the utility function of the packet. For rate-based flows (see below), the utility of a packet is equal to its benefit. For window-based flows the utility function is the benefit rate (benefit per round).

Trades. The objective of each packet is to maximize its utility. Thus, when two packets are paired in a trading period, their inventories and their trading strategies are used to determine if they can agree on a mutually profitable trade, in which one packet offers money and the other offers negative delay. The obvious prerequisite for a trade to take place is that both packets prefer their post-trade inventories to their corresponding pre-trade inventories. For this to be possible, there must be "surplus value" from a potential trade. In this case, both packets can benefit, i.e., increase their utility, if they come to an agreement.

Flow Types and the Cost of Delay. The delay that a packet experiences has a negative impact on its utility. The value is a non-increasing function of the delay. Window-based flows employ a feedback-based mechanism, the congestion window, which determines the maximum number of packets that the flow may have in-flight. Every packet that is in-flight occupies one of the available positions in the congestion window of a window-based flow. The more a packet delays its arrival, the longer the following packet will have to wait to use the occupied window position. Therefore, the impact of packet delays for window-based flows is twofold; the decreased value of the delayed packet and the reduced packet rate. On the other hand, for rate-based flows which submit packets with some given rate, the only consequence due to packet delays is the reduced packet value.

Assume a rate-based packet p with balance α_1 and delay $d_1 < d_{p,\max} - d_\epsilon$, for some d_ϵ. When a trade changes the delay from d_1 to $d_2 = d_1 + d_\epsilon$, then this also changes the value of the packet from $v(d_1)$ to $v(d_2)$. The difference between these two values determines the compensatory price ρ for the packet.

$$\rho = v(d_1) - v(d_2) = v(d_1) - v(d_1 + d_\epsilon) = c_p d_\epsilon \ . \tag{1}$$

At this price, the utility of the packet remains unchanged after the trade. A packet would agree to sell for a price $\rho_s > \rho$, or to buy for $\rho_b < \rho$.

For window-based flows, however, the price estimation needs more attention. Assume a window-based packet with delay $d_1 < d_{p,\max} - d_\epsilon$ and account balance α_1. Before the trade, the utility (benefit rate) is $r_1 = (v_1 + \alpha_1)/d_1$. If the packet agrees to trade its position and to increase its delay by d_ϵ, then the utility is $r_2 = (v_2 + \alpha_2)/d_2$. Then, by setting $r_1 = r_2$ we obtain the compensatory price ρ for the trade.

$$\frac{v_1 + \alpha_1}{d_1} = \frac{v_2 + \alpha_2}{d_2} \Rightarrow \frac{V - c_p d_1 + \alpha_1}{d_1} = \frac{V - c_p(d_1 + d_\epsilon) + (\alpha_1 + \rho)}{d_1 + d_\epsilon} \Rightarrow$$

$$\rho = (V + \alpha_1)\frac{d_\epsilon}{d_1} \ . \tag{2}$$

The above expression for the price ensures that the utility function of the packet remains unchanged. A packet would agree to sell its position, for a price $\rho_s > \rho$,

or to buy a position ($d_\epsilon < 0$) for $\rho_b < \rho$. Unless otherwise specified, the final trading price when a trade takes place will be the average of the ρ_s of the seller packet and ρ_b of the buyer packet. We illustrate the PacketEconomy approach in a representative scenario.

3 Equilibria with Monetary Trades

A Representative Scenario. We examine a simple scenario that produces an interesting configuration. It consists of a set of N window-based flows f_i, for $i \in \{1 \dots N\}$, each with a constant window size w_i, and $\sum_i w_i = q$. When a packet is served by the router it is immediately replaced by an identical packet submitted by the same flow. This is a simplifying but plausible assumption. In reality, when a flow packet arrives at its destination, a small size acknowledgment packet (ACK) is submitted by the receiver. When the sending flow receives the ACK it submits a new identical packet that immediately enters the queue. We assume $b = 1$ trading period per round but in general b can be any integer $b > 0$.

Failure States. For each packet, there is a small probability p_f for an extra delay of d_f rounds, where d_f is a discrete random variable in $\{1, 2, \dots, q-1\}$. These delays correspond to potential packet failures of real flows, and occur between the service of a packet and the submission of its replacement. By convention, the delay d_f is added to the delay of the packet that has just been served. If more than one packets enter the queue at the same time (synchronized due to delays), their order in the queue is decided upon uniformly at random. A packet that does not participate in any trade and does not suffer delay due to failure will experience a total delay of q rounds.

Packet States and Strategies. The state $\tau_p(t)$ of a packet p in round t is a pair $\tau_p(t) = (I_p(t), rel_p(t))$, where $I_p(t)$ is the inventory of the packet and $rel_p(t)$, which is meaningful only in failure states, is the remaining number of failure rounds for the packet. The state of all packets of the economy in round t determines the state of the whole economy $\tau(t) = \prod_{p=0}^{q-1} \tau_p(t)$. From a packet's point of view, a trade is simply an exchange of its inventory state (budget, delay and position) with a new one. Consequently, a pure strategy of a packet is a complete ordering of the possible states of its inventory. In each round, the packets that are waiting move by default one position ahead and, thus, enter a new inventory state. We assume that the packet ignores the impact of its state and strategy on the state of the packet population. In every trading period the packet assumes the same stationary state of the economy.

Definition 1. *Let $\tau(t)$ be the state of the economy in round t.*

Lemma 1. *$\tau(t)$ is an ergodic Markov chain.*

Proof. Assume $b = 1$ trading period per round. In each round, the economy moves to a new state with transition probabilities that depend only on the current state and the strategies of the packets. Let σ_p be a pure strategy of each

packet p of a flow and σ be a pure strategy profile of the whole economy. Then, there is a corresponding transition probability matrix P^{σ} for the economy. Let σ_m be a mixed strategy profile of the whole economy. Then the corresponding transition probability P^{σ_m} of the economy for σ_m is an appropriate convex combination of the transition matrices of the supporting pure strategies. In case of multiple trading periods per round ($b > 1$), the economy makes b state transitions per round.

The number of potential states for a packet is finite and, consequently, the number of states for the whole economy is also finite.

Definition 2. *A zero state τ_0 is a state of the economy in which all packets have zero budget and each packet p has delay $d_p(t_0) = pos_p(t_0) + 1$, where t_0 is the current round of the router.*

Assume that in round t the packet at position 0 fails for $q-1$ rounds, in round $t+1$ the next packet at position 0 fails for $q-2$ rounds etc. Then after q rounds all new packets will simultaneously enter the queue. Each packet will have zero budget and by definition their ordering will be random. This also means that for each packet p, $d_p(t) = pos_p(t) + 1$. Thus, in round $t + q$ the economy will be in a zero state. The probability for this to happen is strictly positive and thus each zero state τ_0 is recurrent. Since the number of states of the economy is finite, the states that are attainable from zero states like τ_0 form a (finite in size) class of irreducible states. Moreover, each zero state is aperiodic, and thus each of the states of the class of attainable states is also aperiodic. It is known that any finite, irreducible, and aperiodic Markov chain is ergodic.

Lemma 2. *(Proof omitted) For each pure strategy profile σ of the economy, there is a unique stationary distribution π_{σ} of the economy.*

An interesting argument which can now be applied is that given the stationary distribution of the economy, each trading period becomes a finite state game.

Lemma 3. *(Proof omitted) For every idle packet, each trading period of the economy corresponds to a finite strategic game.*

Theorem 1. *(Proof omitted) A NE exists where packets perform trades.*

Pipelined Shuffling. A core operation of the PacketEconomy is the random pairing of the packets that takes place in each trading period to generate the trading pairs. We present a new parallel algorithm that can support the random pairing procedure in real time. The new algorithm (Algorithm 1) is a parallel, or better, a pipelined version of the random shuffling algorithm of Fisher-Yates, which is also known as Knuth shuffling [7,22,14]. We call the new algorithm *Pipelined Shuffling*. Its core is a pipeline of q instances $0, 1, \ldots, q-1$ of the Fisher-Yates algorithm. At time t, instance k is at step $t+k \bmod q$ of the random shuffling algorithm.

Theorem 2. *The Pipelined Shuffling algorithm delivers a random shuffle every $O(1)$ parallel time steps on a q processors EREW PRAM.*

Algorithm 1. Pipelined Shuffling

```
1: procedure SHUFFLE(int[] a)
       for i from 0 to q-2 do {
           j = random int in i ≤ j ≤ q − 1;
           exchange a[j] and a[i]}
2: end procedure
1: procedure PARALLELSHUFFLE(int[][] A)
       for i from 0 to q-1 do in parallel {
           processor i: wait for i periods;
           processor i: while (true) {Shuffle(A[i]);}
2: end procedure
```

The Scheduling Problem. The underlying algorithmic problem of the PacketEconomy is a scheduling problem of network packets. From the router's point of view, this problem is a single machine scheduling problem with a max weighted total wealth objective.

Definition 3. *Max-Total-Wealth Scheduling (MTW). A set of n jobs j_i, for $i = 1, \ldots, n$. Job j_i has processing time p_i, release date r_i, deadline d_i and weight w_i. Let c_i be the completion time of job i in a schedule. The objective is to find a non-preemptive schedule that maximizes the total wealth $W = \sum_i w_i \cdot \max(d_i - c_i, 0)$.*

The release date r_i is the time when packet i enters the queue and the deadline d_i is the time when the value of the packet becomes zero. For MTW on a network router the following assumptions hold: a) The queue discipline is work-preserving, meaning a non-empty router is never left idle, b) the number of packets in the queue at any time is bounded by a constant (the maximum queue size), and c) the packet sizes may differ by at most a constant factor. In this work, we assume that all packets are of the same size.

The complexity of the MTW problem depends on the assumptions made. It is not hard to show that without deadlines, even the online version of MTW can be optimally solved; there is a 1-competitive algorithm for MTW without deadlines [5]. Moreover, MTW with deadlines can be offline solved in polynomial time as a linear assignment problem.

However, due to the on-line nature and the finite queue size of the PacketEconomy router, the above conventional scheduling algorithms do not seem to naturally fit the MTW problem of the PacketEconomy. Especially for window-based flows, where packet transmission is a closed loop, the order in which the queued packets are served influences, if not determines, the next packet that will enter the queue. Thus, even the online assumption may not be appropriate. A different approach to study the scheduling problem of the PacketEconomy is to consider the (average) packet rate of the flows, as shown in the following example.

Example 1. Assume a scenario with window-based flows and 5 economy packets and 5 business packets. There is a deadline of 40 rounds on the maximum delay of

the economy packets. Moreover, all business packets have to be treated equally. The same holds for the economy packets. Consider the scenario where each economy packet will be served with a rate of 1/40 packets/round and delay of 40 rounds and the business flows share the remaining bandwidth; each business packet is served at a rate of 7/40 packets/round and delay 40/7 rounds. This is an upper bound on the rate of total wealth for the router for this scenario.

4 The Effect of Trades

The NE of the representative scenario shows that, in principle, money can be used at a microeconomic level to coordinate network packets. By definition, the flows of the scenario can only benefit through the use of money; each trade is a weak Pareto improvement for the current state of the economy. In this section we further examine the effect of trades.

In the PacketEconomy, each packet can increase its utility by making trades. To show the potential of the approach, consider a packet of maximum priority that pays enough to make any trade that reduces its delay. In the analysis, we will assume that the probability of packet failures is very low, and thus ignore it. We focus on window-based flows, present an exact calculation for the average delay of this packet and then derive simpler, approximate bounds.

Lemma 4. *The average delay $E[d]$ of the packet is*

$$E[d] = \sum_{s=1}^{q} s \left(\frac{1}{q-2}\right)^{s} (s-1) \frac{(q-2)!}{(q-s-1)!} \,. \tag{3}$$

Proof. Let $rand(L, U, s)$ be a uniformly random integer in $\{L, L+1, ..., U\} \backslash \{s\}$ and $pos(p)$ the current position of packet p. Then, the probability $Pr[d > s]$ is

$$= \prod_{k=1}^{s} Pr[rand(1, q-1, pos(p)) \geq s-k+1] = \frac{q-s-1}{q-2} \cdot \frac{q-s}{q-2} \cdots \frac{q-2}{q-2} \Rightarrow$$

$$Pr[d > s] = \left(\frac{1}{q-2}\right)^{s} \cdot \frac{(q-2)!}{(q-s-2)!} \text{ , and}$$

$$Pr[d = s] = Pr[d \leq s] - Pr[d \leq s-1] = \left(\frac{1}{q-2}\right)^{s} (s-1) \frac{(q-2)!}{(q-s-1)!}.$$

Applying the definition of the expected value completes the proof.

Lemma 5. *(Proof omitted). Let $X^{d}_{\min}$ be the minimum of $n > 0$ discrete uniform random variables (RV) in $[L, U]$ and $X^{c}_{\min}$ be the minimum of n continuous uniform RV in $[L, U]$. Then*

$$E[X^{d}_{\min}] \leq E[X^{c}_{\min}] \leq E[X^{d}_{\min}] + 1 \,. \tag{4}$$

Lemma 6. *The average delay of the packet does not exceed* $\frac{-1+2b+2\sqrt{2b(q-2)}}{2b}$. *For* $b = 1$ *the bound is* $\frac{1}{2} + \sqrt{2(q-2)}$.

Proof. A packet that enters at position $q-1$ has been served when it advances at least q positions. Note that each random trading partner corresponds to a uniform random number in $[1, q-1]$. To admit a more elegant mathematical treatment we prefer the continuous distribution. Lemma 5 makes this possible.

Assume that a packet has just entered the queue at position $q-1$. Let b be the number of trading periods per router round. Assume that the packet spends at least k rounds in the queue until it reaches position 1. During these k rounds the packet will make bk random draws and will make k single position advancements. From Lemma 7 we obtain that the average value of the minimum of the bk random draws is

$$\frac{1}{bk+1}(q-2) \; .$$

Lemma 7. *(Proof omitted) Let* $X_1, X_2, \ldots, X_k$ *be continuous uniform random variables in* $[0, U]$ *and let* $X_{\min} = \min_{i=1,\ldots,k} X_i$. *Then* $E[X_{\min}] = \frac{1}{k+1}U$.

Note that the average number of rounds and draws until it achieves its best draw is $(k+1)/2$ and $(bk+1)/2$, respectively. We will add one to the value of the average minimum draw, because the minimum position that can be traded is position 1. Position 0 is the one that is currently being served.

Now, assume that after the k rounds and bk draws the packet advances for h additional rounds until it reaches position 1. From position 1 it needs a final round to proceed to position 0 and be served. Thus, the total delay of the packet is $k+h+1$, and

$$\frac{1}{bk+1}(q-2) + 1 - (k+1)/2 - h - 1 \leq 0 \; .$$

We solve for k and obtain that the larger of the two roots of k is

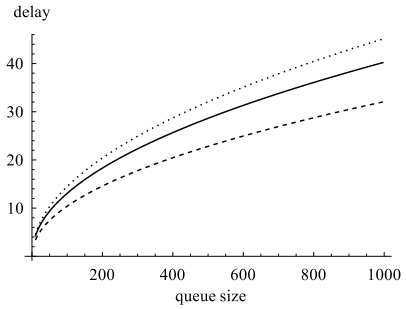
$$k = \frac{-1-b-2bh+\sqrt{(1+b+2bh)^2+4b(2q-5-2h)}}{2b} \; . \tag{5}$$

The total delay $k+h+1$ is minimized at $h = (1-b)/(2b)$. Substituting $h = (1-b)/(2b)$ in Equation 5 gives that the minimum value of $k+h+1$ is

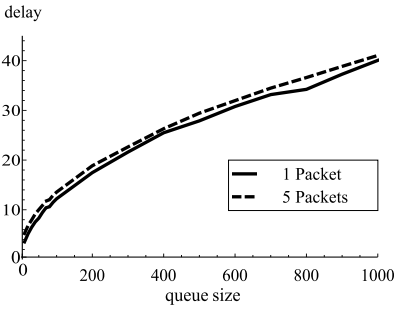
$$k+h+1 = \frac{b-1+2\sqrt{2b(q-2)}}{2b} \; .$$

The average delay cannot be larger then the above value. This completes the proof of Lemma 6.

The above lemma can be generalized to the case where only one packet in every $c > 0$ packets in the queue is ready to sell its position. We simply assume b/c trading periods per round. Then, the average delay of the business packet is not larger than $1 - (c/2) + \sqrt{2c(q-2)}$. Similarly, we can show:



(a) The exact average delay (inner line) and the lower and upper bounds on the average delay



(b) Experimental measurement of the delay for the cases of one business packet and five business packets.

Fig. 4. Delay of the business packet with respect to the queue size

Lemma 8. *(Proof omitted) The average delay of the packet is at least* $(-1 + 2b + \sqrt{1 - 8b + 4bq})/(2b)$. *For* $b = 1$ *the bound is* $(1/2) + \sqrt{4q - 7}$.

This lemma too, can be generalized to the case where only one packet in every $c > 0$ packets in the queue is ready to sell its position. In this case the average delay of the business packet is not less than $\frac{1}{2}(2 - c) + \frac{1}{2}\sqrt{c^2 - 8c + 4qc}$.

In Figure 4, analytical and experimental results for the delay of the business packet are presented. In the long version of this work [5], we use the lemmas of this section to analyze the packet delays and the social wealth of a PacketEconomy instance for the cases of no trades, ideal trades, and PacketEconomy trades.

5 Conclusion

We presented an economy for network packets and showed the existence of NE where money circulates to the benefit of the flows. The basic computational step of the PacketEconomy can be executed in $O(1)$ parallel time on fairly simple multi-core hardware, making it appropriate for modern network router demands.

There are several other issues that have to be addressed for such a model to be of practical importance. For example, a greedy flow may submit economy packets to the network simply to collect money. A realistic economic model has to anticipate such scenarios and address them with appropriate rules. One approach could be to have the router restricting the final budget of any packet to be non-positive, or more effectively, impose router-entry costs on every packet depending on the current load.

Overall, we examined how money can be used at a microeconomic level as a coordination tool between network packets and we believe that our results show that the PacketEconomy approach defines an interesting direction of research for network games. We are currently examining the use of fiat money and the implemention of the PacketEconomy in a realistic network context.

Acknowledgements. The first author, is grateful to Paul Spirakis for inspiring discussions on the intersection of Algorithms and Game Theory. We also wish to thank Vasilis Tsaoussidis for insightful discussions on using budgets in Internet router queues.

References

1. Akella, A., Seshan, S., Karp, R., Shenker, S., Papadimitriou, C.: Selfish Behavior and Stability of the Internet: a Game-Theoretic Analysis of TCP. In: SIGCOMM 2002, pp. 117–130 (2002)
2. Chiu, D.-M., Jain, R.: Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. Comp.Netw.ISDN 17(1), 1–14 (1989)
3. Cole, R., Dodis, Y., Roughgarden, T.: How much can taxes help selfish routing? In: EC 2003, pp. 98–107. ACM (2003)
4. Cole, R., Dodis, Y., Roughgarden, T.: Pricing network edges for heterogeneous selfish users. In: STOC 2003, pp. 521–530. ACM (2003)
5. Efraimidis, P.S., Koutsiamanis, R.-A.: On money as a means of coordination between network packets. CoRR, abs/1208.3747 (2012)
6. Efraimidis, P.S., Tsavlidis, L., Mertzios, G.B.: Window-games between TCP flows. Theoretical Computer Science 411(31-33), 2798–2817 (2010)
7. Fisher, R.A., Yates, F.: Statistical tables for biological, agricultural and medical research, 3rd edn. (1948)
8. Gao, X., Jain, K., Schulman, L.J.: Fair and efficient router congestion control. In: SODA 2004, pp. 1050–1059 (2004)
9. Gibbens, R.J., Kelly, F.P.: Resource pricing and the evolution of congestion control. Automatica 35, 1969–1985 (1999)
10. Gintis, H.: A markov model of production, trade, and money: Theory and artificial life simulation. Comput. Math. Organ. Theory 3(1), 19–41 (1997)
11. Gintis, H.: Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction. Princeton University Press (2000)
12. Karp, R., Koutsoupias, E., Papadimitriou, C., Shenker, S.: Optimization problems in congestion control. In: FOCS 2000, p. 66. IEEE Computer Society (2000)
13. Kiyotaki, N., Wright, R.: On money as a medium of exchange. Journal of Political Economy 97(4), 927–954 (1989)
14. Knuth, D.E.: The Art of Computer Programming, 2nd edn. Seminumerical Algorithms, vol. 2. Addison-Wesley Publishing Company (1981)
15. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
16. MacKie-Mason, J.K., Varian, H.R.: Pricing the internet. In: Public Access to the Internet, pp. 269–314. Prentice Hall (1993)
17. McKnight, L.W., Bailey, J.P. (eds.): Internet Economics. MIT Press (1997)
18. Odlyzko, A.: Paris metro pricing for the internet. In: EC 1999, pp. 140–147. ACM (1999)
19. Papadimitriou, C.: Algorithms, games, and the internet. In: ACM STOC 2001, pp. 749–753 (2001)
20. Shenker, S.J.: Making greed work in networks: a game-theoretic analysis of switch service disciplines. IEEE/ACM Trans. Netw. 3(6), 819–831 (1995)
21. Smith, A.: An Inquiry into the Nature and Causes of the Wealth of Nations. Project gutenberg ebook edition (1776)
22. Wikipedia. Fisher-Yates shuffle (entry) (2011) (accessed February 06, 2011)