Performance Evaluation of τ-AIMD over Wireless Asynchronous Networks

Adrian Lahanas
Dept. of Computer Science
University of Cyprus
Nicosia, Cyprus

Vassilis Tsaoussidis
Dept. of Electrical Engineering
Demokritos University
Xanthi, Greece

Abstract

The work describes the performance of two congestion control algorithms: AIMD and τ-AIMD. The first is the default mechanism of TCP; the second is a proposed congestion control algorithm that improves fairness of AIMD. We consider asynchronous networks where TCP flows have different propagation delays, a portion of their link is wireless, and they compete for resources over a single bottleneck link. We show that τ-AIMD improves the performance of flows that have long propagation delay and the fairness of the network. In wireless links τ-AIMD outperforms AIMD and the cost of lost packets (or wasted energy) is the same as that of AIMD.

1 Introduction

Additive Increase/Multiplicative Decrease (AIMD) is the algorithm that controls congestion in the Internet [6, 11]. It is coded into TCP and adjusts its sending rate mechanically, according to the 'signals' TCP gets from the network. A lost packet from the data it pumps into the network is considered as a congestion event and therefore, AIMD decreases the sending rate of TCP. Otherwise, when data is transmitted successfully, the AIMD increases the sending rate of TCP by a packet per RTT (round trip time). It is proved that these adjustments bring the network load into an equilibrium and TCP flows converge into a fair state.

Fairness of TCP is an over-argued topic because of the nuances of fairness [4, 12]. In this work we support the *max-min* notion of fairness where all flows get the same treatment (consume the same resources) from the networks. Max-min fairness is discussed in detail in [4]. Judged from the max-min notion of fairness, in asynchronous networks, TCP does not share network resources equally among flows [6, 9]. The problem is caused by short RTT flows which increase their sending rate faster than long RTT ones. Short RTT flows consume more resources from the network, leaving thus less resources for the long RTT flows.

In wireless environments where link disconnections or bit errors are more frequent than in wired networks, the problem of fairness becomes more apparent. In these environments flows experience more packet drops, which they consider as congestion signals, and reduce their sending rate more frequently. Long RTT flows are those that experience the most severe treatment from the network. If we add the wireless link error factor to the limited resources the short RTT flows leave in the system, it is obvious that these flows operate at very low transmission rates and their process of recovery is very time-consuming.

Although fairness of TCP over heterogeneous (wired and wireless) asynchronous networks is an open issue, it has not received enough attention. A solution or an improvement of the problem is important for two reasons: *i*) it will speed-up the recovery process of long RTT flows when multiple errors occur in the network[1], *ii*) when long RTT flows recover faster and have rates close to short RTT flows, they utilise better the network resources or the available bandwidth at the wired and at the wireless portion of the network.

Performance of TCP over wireless links has been studied in a range of works (e.g. [19, 7, 8]) and several mechanisms have been proposed to improve performance over such links. However, the focus of these works has been the wireless hop of the network rather than the global system: flows with heterogeneous RTTs that traverse wired or wireless links. In this work we experiment with a congestion control algorithm, named τ-AIMD, whose purpose is to improve fairness of TCP flows in asynchronous networks. This algorithm derives from AIMD but the rate of the protocol is increased proportionally to its RTT (whereas in TCP it increases a packet per RTT). The long RTT τ-AIMD flows increase their rate faster than they do with AIMD and consume more resources. This scheme however, does not achieve max-min fairness among flows. The experiments in this work are focused mainly on the fairness achieved by AIMD and τ-AIMD in asynchronous networks where a portion of the link is wireless.

The rest of this work is organised as follows: Section 2 describes the related work on fairness and Section 3 gives detailed description of AIMD and τ-AIMD mechanism; Section 4 describes the experimental set-up and Section 5 evaluates the experimental results. Section 6 summarises the work.

2 Related Work

The AIMD was first introduced in [6] where the authors studied the algorithm in terms of fairness and convergence to equilibrium. Jacobson in his work [11] proposed and coded into TCP a mechanism similar to AIMD and studied it in terms of congestion control. The AIMD algorithm is effective in congestion control but, as its authors have stated out, in asynchronous networks the flows do not get the same share [6]. The concept of AIMD itself is quite simple, general and very effective. Although the same concept can be used in different ways to improve a specific performance characteristic of the network (e.g. [9, 13, 14, 2]), the AIMD of TCP is limited into achieving only equilibrium in the network.

The proposal in [9] considers the fairness problem of TCP in asynchronous networks. Increasing the sending rate of TCP by $a \cdot r^2$ (where a is a constant and r is the RTT of the flow) can achieve the same rate among the flows and improve fairness. This mechanism is based on AIMD principles but is shown experimentally that the overhead (or the number of lost packets) increases sig-

---

[1] Multiple errors might result in sequential decreases of the transmission rate of the flow, which implies a few packet in transit. When in-transit packets are scarce a loss is very costly because it could result in time-outs or might make useless the 3 DACK mechanism of TCP

nificantly [10]. Choosing the parameters of this mechanism (i.e. constant a) is still an open problem. Analysis of congestion avoidance of TCP in asynchronous systems has shown that when it reaches an equilibrium, the formula

$$F_A^h = \sum_{i \in S} \frac{1}{\tau_i} \log \frac{x_i}{a_I + b_D x_i} \tag{1}$$

where x_i are the rates of each flow and τ_i is the RTT of flow i, is maximized [18]. This is also the fairness characteristic of AIMD in asynchronous systems.

The effect of wireless errors on the throughput performance of TCP has been studied and reported in a series of works (e.g. [20]). The work in [17] reports the effect of the wireless errors in the fairness of TCP. The work in [16] reports the fairness of TCP over 802.11 wireless networks. All these works deal with the performance of TCP over wireless local area networks which is a mere part of a large scale asynchronous network (like Internet). In [5] and [3] is studied the performance of TCP over wireless wide area networks. The improvements that the authors propose deal mainly with the throughput of TCP and are limited at the wireless hop of the network.

3 AIMD and τ-AIMD

AIMD is a distributed algorithm that runs at the transport layer of each end-node with a minimum assistance from the network. The main goal of this algorithm is to control the sending rate of each end-node such that an equilibrium can be reached in the network. Accordingly, each user increases its sending rate linearly when resources are available and decreases exponentially the sending rate as soon as the network is congested. The increase and decrease of the sending rate in TCP is controlled by a parameter called congestion window. This parameter records how many bytes TCP has in transit (i.e. in the network). When the whole window is transmitted successfully TCP increases its window by one packet (i.e. one packet per RTT) and when a packet is lost from the window it decreases to half the congestion window. In addition to AIMD TCP has other mechanisms that assist in congestion avoidance process (e.g. ACK clocking and time-outs) or mechanisms that assist in the recovery process after packets loss (e.g. three DACK, Fast Retransmit, Slow Start, Fast Recovery).

Algorithmically the AIMD can be expressed with the following lines:

```
AIMD()
$a_i$ : constant = packet-size()
$W$ : integer // congestion window
repeat forever
    send $W$ bytes in the network
    receive ACKs
    if $W$ bytes are ACKed
        $W \leftarrow W + a_i$
    else
        $W \leftarrow \frac{W}{2}$
end
END-AIMD
```

This continuous process of AIMD achieves equilibrium but, does not achieve max-min fairness in asynchronous networks.

τ-AIMD is designed for asynchronous networks: flows might have different RTT or propagation delays (PD). τ-AIMD controls congestion by using the same increase/decrease mechanism of AIMD. To improve fairness τ-AIMD increases the rate of the flows proportionally to the RTT or PD of the flow. Instead of adding a packet every RTT to the window (a_i in the pseudo-code), τ-AIMD adds as many as is the flow's RTT (i.e. $\tau \cdot a_i$, where τ is the RTT of the flow). Algorithmically, τ-AIMD can be expressed by the following lines:

```
τ-AIMD()
$a_i$ : constant = packet-size()
$W$ : integer // congestion window
repeat forever
    send $W$ bytes in the network
    receive ACKs
    $\tau \leftarrow$ Window-transmission-time()
    if $W$ bytes are ACKed
        $W \leftarrow W + \tau \cdot a_i$
    else
        $W \leftarrow \frac{W}{2}$
end
END-τ-AIMD
```

It is proved in [14] that an asynchronous system where flows use the τ-AIMD as their congestion control algorithm reaches an equilibrium and a window based fairness (i.e. windows of the flows become equal when they converge to their fair state). Window based fairness does not mean that flows achieve max-min fairness. Nevertheless, window based fairness is closer to max-min fairness than F_A^h fairness of AIMD is.

4 Experimental Methodology

We have implemented τ-AIMD into TCP and validated its performance on Ns2 [21] simulator. We use the smoothed RTT (*srtt* variable of Ns2's TCP) to assign values to the τ parameter. Two different sets of experiments are used to validate the performance of the algorithm. In the first set of experiments we set-up a topology with a single bottleneck router A as shown in figure 1. Flows have different propagation delays ranging from 19ms to 129ms and a portion of their link is wireless (from point B to C in each topology). To simulate the wireless link errors we use the On/Off error model presented in [1]. This is a Markov chain model. The packet error rates (PER) in each experiment ranged from 0.01% to 10%.

In the second set of experiments we used a 'linear topology' as that in figure 2. Flows have different PDs and traverse a number of bottleneck routers depending on their PD (e.g. flow F1 traverses 4 bottleneck routers, flow F2 traverses 3, flow F3 traverses 2, and flow F4 traverses 1). In these experiments we were interested in the performance of each algorithm when they traverse a number of bottlenecks and a portion of their link is wireless.

There are four TCP flows in each experiment (four senders and 4 receivers) whose performance we monitored. In addition to these flows we added other background flows at points A and B of each topology that generate forward and

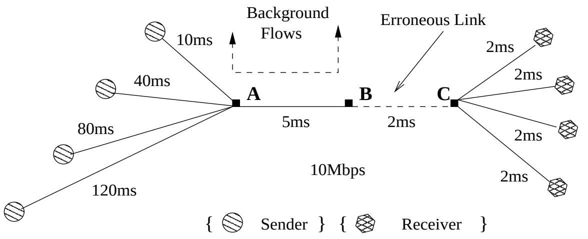

Figure 1: Single Bottleneck Topology.

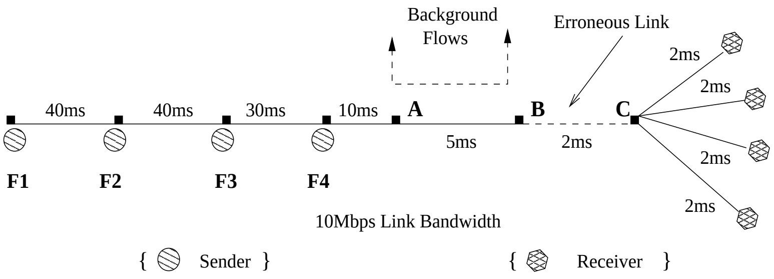

Figure 2: Linear Topology.

backward traffic in order to avoid window and ACK synchronisation. We don't measure the performance of these flows.

The experiments are limited to four flows in order to allow each TCP to expand its window and control the rate mainly with AIMD or τ-AIMD rather than with other mechanisms of TCP (i.e. time-outs and Slow Start). The bottleneck routers use the RED drop policy and are configured with the default Ns2 parameters.

FTP application was attached to each TCP and their task was to send continuously data for a period of 60 seconds - enough to validate the performance of each mechanism. We use *goodput* as a metric to compare the performance of each algorithm. Goodput is defined as the amount of data received at the application over the transmission time. For each PER we conducted 35 experiments and report the average goodput achieved by each flow.

5 Simulation Results

We evaluate first the performance of both algorithms in a network where the only cause of packet loss is congestion and then compare their performance in the presence of errors similar to wireless environments. In figure 3 is displayed the goodput performance of AIMD and τ-AIMD in the single bottleneck topology. The figure shows that when the propagation delay of the flows increases, its goodput performance decreases. The goodput of τ-AIMD flows is higher than the goodput of AIMD when the propagation delay increases whereas, when flows have small propagation delay, the goodput of τ-AIMD is slightly lower. The reason is that long RTT τ-AIMD flows open their window faster than AIMD flows. Therefore, they grab more resources from the network and leave less for the short RTT flows. If the network has less resources the performance of the flows will be lower. In other words, the goodput loss from short RTT flows is converted in goodput gain from long RTT flows. Nevertheless, the fairness performance of the system improves. The figure shows that the goodput of the flow that has delay 80ms from the bottleneck router improves by 16% and the goodput of the flow that has delay 120% ms from the bottleneck improves by 24%.

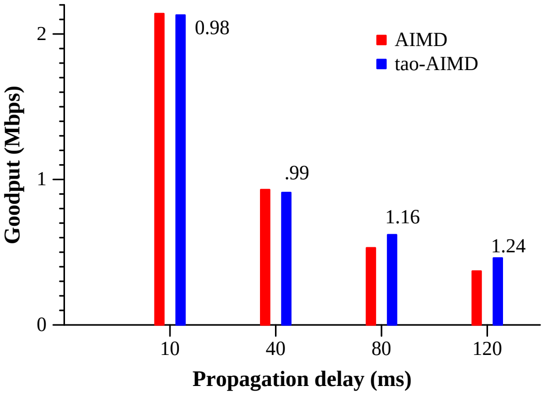

Figure 3: Goodput of TCP-SACK with AIMD and τ-AIMD congestion control algorithms. The numbers on top of each bar indicate the ratio of τ-AIMD over AIMD goodput.

Figure 4 shows the performance of the algorithms in the linear topology with four bottlenecks and in the absence of wireless errors. The performance of AIMD is similar to its performance in the single bottleneck topology and the gain of long RTT τ-AIMD flows over AIMD flows is almost the same. These experiments show the capability of τ-AIMD in improving the fairness and achieving congestion control in asynchronous networks.

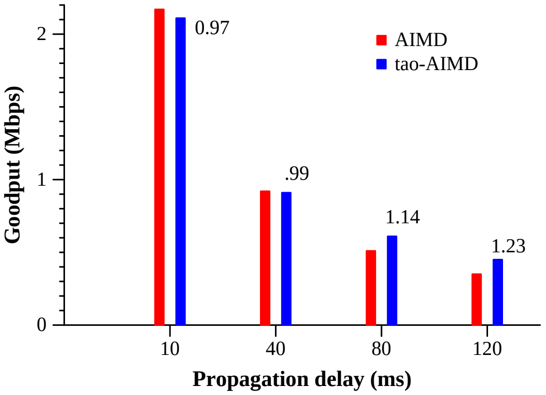

Figure 4: Goodput of TCP-SACK with AIMD and τ-AIMD in the four bottleneck topology. The numbers on top of each bar indicate the ratio of τ-AIMD over AIMD goodput.

In figure 5 is plotted the goodput performance of flows that have delay 10ms and 40ms respectively, from the bottleneck router. The performance is given as a function of the error rate in the wireless channel. When the error rate increases above 0.1% the performance of the flows drops significantly. However, there is not a clear distinction in the performance of each algorithm. The τ-AIMD performs the same as AIMD for short RTT flows because of its implementation in Ns2. The value of 'τ' in the increase formula is calculated by dividing the smoothed RTT variable of TCP by 64, which is almost or bigger than the RTT of the plotted flows[2]. Therefore, τ-AIMD increases its window exactly as AIMD and has the same performance.

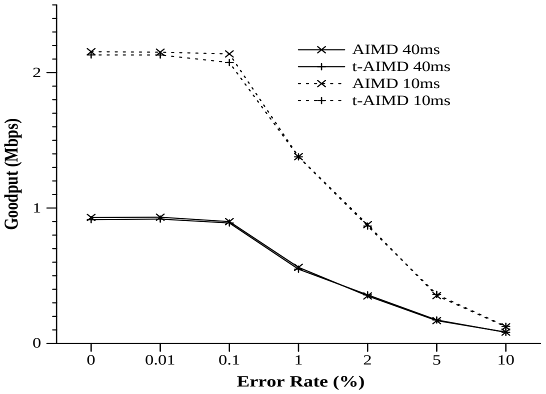

Figure 5: Goodput of flows when the propagation delay is 10ms vs. goodput when propagation delay is 40ms. Goodput is a function of 'wireless' error rate.

When the propagation delay increases the τ-AIMD increases the goodput of long RTT flows. Figures 6 and 7 show that even in the presence of errors characteristic to wireless links, the goodput of τ-AIMD is higher that that of AIMD. This means that the increase mechanism of τ-AIMD benefits also the recovery process of flows after a wireless error event. Expanding faster the window makes the flow exploit better the available bandwidth. When the packet error rates increases up 10% the goodput performance of AIMD and τ-AIMD becomes almost equal. In such error rates the flows probably experience loss of the whole window and result in time-outs or in minimal window values. As a result, the traffic in such high error rates is shaped by time-outs and Slow Start mechanisms rather than AIMD and τ-AIMD.

In the four bottleneck linear topology the performance of AIMD and τ-AIMD is similar to the performance in the single bottleneck topology. Because of the similarity we display only the results with the experiments where flows have delay 10ms and 120ms respectively, from the bottleneck router A. These results are shown in figure 8.

In figure 9 is displayed the *overhead* or the ratio of loosed packets over the received data at the application. It is obvious that the loss increases when the error rate at the wireless portion of the link increases. The experiments show that the overhead of both mechanisms is approximately the same. In erroneous wireless networks the overhead is mainly due to the loss at the wireless portion of the network rather than loss at the bottleneck congested routers. For both

---

[2] When the division is equal to zero, we set τ to 1. The source code of the algorithm is available on line at http://www.cs.ucy.ac.cy/~ladrian/software.html.

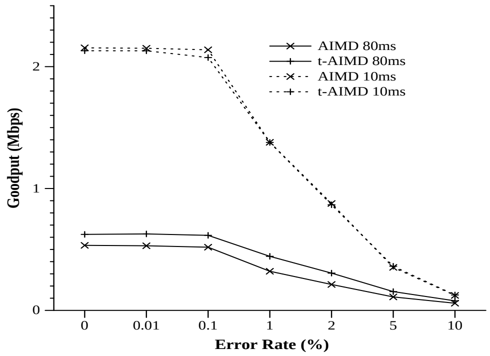

Figure 6: Goodput of flows when the propagation delay is 10ms vs. goodput when propagation delay is 80ms.

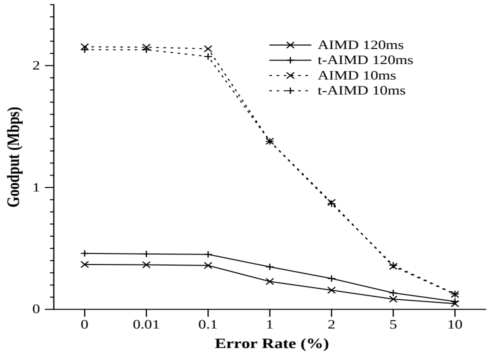

Figure 7: Goodput of flows when the propagation delay is 10ms vs. goodput when propagation delay is 120ms.

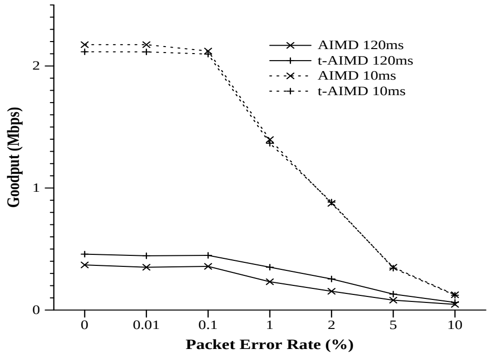

Figure 8: Goodput of flows when the propagation delay is 10ms vs. goodput when propagation delay is 120ms, in the four bottleneck topology.

protocols the overhead increases almost linearly with the error rate. Since the overhead of both protocols is almost equal we report the overhead of flows that have delay 10ms and 120ms from the bottleneck router A. In the four bottleneck topology these numbers are similar.

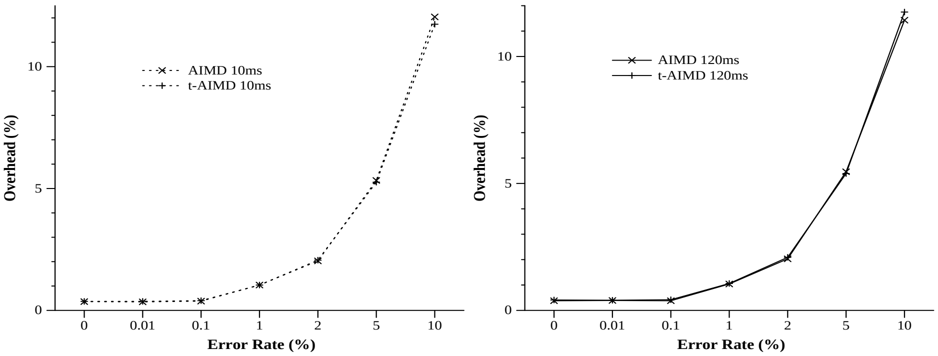

Figure 9: Left: Overhead of flows when propagation delay is 10ms. Right: Overhead of flows when propagation delay is 120ms.

An important conclusion that we infer from these experiments is the gain of τ-AIMD flows over the AIMD ones. In asynchronous (wired or wireless) networks the goodput of τ-AIMD flows is higher than the goodput of AIMD flows. Furthermore, the overhead or lost energy of each protocol is the same. This means that τ-AIMD can achieve higher goodput than AIMD at the same wasted energy cost. This feature of τ-AIMD is important for wireless devices where energy is limited and transmission time effect their battery life.

6 Conclusions

The work studies experimentally the performance of τ-AIMD and AIMD over asynchronous networks. We have shown the potential of τ-AIMD to improve fairness of the network even when a portion of it is wireless. The goodput of long RTT flows is increased whereas the goodput of short RTT flows reduces by the same portion. Although the goodput of long RTT flows increases, the cost of energy loss (or packet loss) is the same as the cost with AIMD. This makes τ-AIMD a protocol suitable for wireless devices where energy is limited and its loss effects the battery time of the device. τ-AIMD flows transmit an amount of data faster than AIMD and at a lower energy cost.

References

[1] A. Abouzeid, S. Roy, and M. Azizoglou. Stochastic Modeling of TCP over Lossy Links. *INFOCOM 2000*, March 2000.

[2] P. Attie, A. Lahanas, and V. Tsaoussidis. Beyond AIMD: Explicit Fair-share Calculation. In *Proceedings of the ISCC'03*, June 2003.

[3] H. Balakrishnan, V. Padmanabhan, and R. Katz. The Effects of Asymmetry in TCP Performance. In *Proceedings of the 3rd ACM/IEEE Mobicom Conference*, pages 77–89, September 1997.

[4] D. Bertsekas and R. Gallager. *Data Networks*, chapter 6, pages 493–530. Prentice Hall, 1987.

[5] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt. Flow Aggregation for Enhanced TCP over Wide-Area Wireless. In *Proceedings of the INFOCOM '03*, March 2003.

[6] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[7] A. Chockalingam, M. Zorzi, and R. Rao. Performance of TCP on Wireless Fading Links with Memory. In *Proceedings of the IEEE ICC'98, Atlanta, GA*, pages 201–206, June 1998.

[8] V. Tsaoussidis *et al.* Energy / Throughput Tradeoffs of TCP Error Control Strategies. In *Proceedings of the 5th IEEE Symposium on Computers and Communications, ISCC*, pages 150–156, July 2000.

[9] S. Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks Part1: One-way Traffic. *ACM Computer Communication Review*, 21(5):30–47, October 1991.

[10] T. H. Henderson, E. Sahouria, S. McCanne, and R. H. Katz. On Improving the Fairness of TCP Congestion Avoidance. In *Proceedings of the IEEE Globecom*, 1998.

[11] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of the ACM SIGCOMM '88*, pages 314–329, August 1988.

[12] F. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[13] A. Lahanas and V. Tsaoussidis. Exploiting the Efficiency and Fairness Potential of AIMD-based Congestion Avoidance and Control. *Journal of Computer Networks, COMNET*, 43:227–245, 2003.

[14] A. Lahanas and V. Tsaoussidis. τ-AIMD for Asynchronous Networks. In *Proceedings of the ISCC'03, Antalya, Turkey*, June 2003.

[15] T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5:336–350, June 1997.

[16] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha. Understanding TCP Fairness over Wireless LAN. In *Proceedings of the INFOCOM '03*, March 2003.

[17] D. Vardalis and V. Tsaoussidis. On the Efficiency and Fairness of Congestion Control Mechanisms in Wired and Wireless Networks. *The Journal of Supercomputing, Kluwer Academic Publishers*, 23, November 2002.

[18] M. Vojnovic, J.Y. Le Boudec, and C. Boutremans. Global Fairness of Additive-Increase and Multiplicative-Decrease with Heterogeneous Round-Trip Times. In *Proceedings of the IEEE INFOCOM'00*, pages 1303–1312, March 2000.

[19] G. Xylomenos and G. Polyzos. TCP and UDP Performance over a Wireless LAN. In *Proceedings of the IEEE INFOCOM*, pages 439–446, March 1999.

[20] M. Zorzi and R. Rao. Perspective on the Impact of Error Statistics on Protocols for Wireless Networks. *IEEE Personal Communications Magazine*, 6:34–39, October 1999.

[21] "—" The Network Simulator - NS-2. Technical report, Web Page: http://www.isi.edu/nsnam/ns/. Version 2.27, January 2004.