Measuring TCP over WiFi: A Real Case

Mirko Franceschinis[1], Marco Mellia[2], Michela Meo[2], Maurizio Munafò[2]
[1]Istituto Superiore Mario Boella - Torino - Italy
[2]Dipartimento di Elettronica - Politecnico di Torino - Italy

***Abstract*— We present a measurement study of an operative WLAN. After briefly presenting the methodology used to generate traffic and measure performance indices, we present results focusing on the performance perceived by TCP connections, to understand the mechanisms which have a major effect on performance. We investigate the impact of TCP parameters, like the maximum congestion window, the effectiveness of link layer mechanisms such as RTS/CTS, and multirate transmissions. Rather than only presenting aggregate measures, we show results relative to individual TCP connections, considering throughput as well as TCP round trip time, congestion window and segment retransmissions.**

I. Introduction

Wireless Local Area Networks (WLAN) based on 802.11b/g technology (commercially known as WiFi) have become, in the last few years, quite popular and widespread. From a QoS and traffic management point of view, the main problem with WiFi networks is the relative low capacity of the shared radio channel. In a work environment used to FastEthernet connectivity, in which large amount of data is exchanged among the terminals, e.g., document distribution in a corporate intranet, the 11Mbps (gross and shared) offered by the 802.11b technology may appear as a severe limitation. A second crucial problem is the performance reduction of data transfer using TCP in wireless network scenarios. Indeed, the nature of the radio channel and the random access to the shared resource cause variable packet delay and loss rate, which are the main triggers of the TCP congestion control mechanism. However, since most of the Internet user applications (web browsing, FTP data transfer, email services) uses TCP for reliable data transfer, the degradation of TCP performance is perceived by the user as a QoS degradation of the wireless network.

The objective of this paper is to evaluate the performance of TCP connections over WiFi, from a common user point of view, by considering an operative WLAN. By loading the wireless network with synthetic traffic, we stress the system and measure the TCP performance through a tool, named *Tstat* [1], that was developed at the Politecnico di Torino and specifically designed to evaluate TCP performance metrics. This allows us to focus on the QoS received by individual TCP connections rather than on the performance of the wireless network as a whole. In our evaluation, we consider various protocol settings, at both the TCP and the link layer, such that we can investigate the interaction between the TCP congestion control and the WLAN MAC, and identify the most critical system configurations.

This work was partly supported by the Italian Ministry for Education, University and Research under project TANGO, and by the European Community under the NoE Newcom. Dr. Franceschinis was supported by the ISTI-CNR during this project.

The problem of TCP behavior in wireless environment is well known and was quite extensively studied in the literature, see [2] and references therein for a good overview. Despite our approach is similar to the one followed in previous work, the use of a tool that was specifically developed for the analysis of the behavior of TCP connections allows us to investigate some parameters, such as the round trip time, the number of retransmitted segments, the value of the TCP congestion window size, which, at the best of our knowledge, were not considered in previous studies. These parameters, together with other more traditional ones, such as throughput, indicate what kind of service TCP receives from the WiFi access network below. In particular, the main conclusions we learn from our experiments are the following:

- In scenarios with most of the traffic on the downlink, the use of large values of the TCP maximum congestion window induces high variance on the performance of TCP connections and some unfairness due to packet drops at the AP queue. When the access network is the bottleneck, limited congestion window may reduce unfairness.
- Differently from what was observed in [3], [4], in our scenarios with uplink traffic, RTS/CTS mechanism improves TCP throughput; indeed, despite the higher overhead, by reducing collisions, RTS/CTS reduces TCP retransmission probability too, and this translates into higher throughput.
- Due to different channel conditions, some severe unfairness may appear in the uplink direction for stations which receive, in the downlink direction, a service which is almost as good as the one of other stations. This severe unfairness cannot be eliminated by neither changing the transmission rate, nor introducing the RTS/CTS mechanism.

A further contribution of the paper consists in presenting a measurement environment which, despite being extremely easy and cheap to implement, allows sophisticated analysis of TCP behavior in WLANs. The measurement environment consists of some common hardware and some code which can be freely downloaded from our site [5].

II. Network Testbed

A. Traffic generation and measurement tool

Our setup is based on two elements: the traffic generator tool and the measurement tool.

The traffic generator is quite general and allows the creation of a wide set of traffic scenarios. The basic setup consists of one *master* station running the server side application, and an arbitrary number of *slave* stations running the client version of the application. The master controls the test scenario, which is described by simple text files. During a test, the master acts as a *data server*, while all slaves act as *data clients*, which download data from the server.

The interaction between slaves and master consists of two phases: a registration phase followed by a download phase. During the registration phase, each slave registers to the master, receiving a traffic profile. The traffic profile contains information about three traffic parameters:

- N_t - number of trials: the number of data transfers requested by the client to the server;
- D - datasize: the size of each requested data transfer;
- T_{off} - offtime: the time between the end of a data transfer and the next data request.

A stochastic description of the parameters listed above allows us to create flexible traffic scenarios. During the registration phase, the master instructs all the slaves about the time at which the download phase will start, therefore synchronizing the test startup time.

Using a single TCP connection, the client requests a given amount of data and performs the correspondent download. After a time interval equal to T_{off}, it then performs another request until all the N_t trials have been performed. To avoid perfect synchronization among clients, the beginning of the download phase is randomly delayed at each client.

As measurement tool we employ Tstat, which provides the collection and the statistical analysis of TCP/IP traffic, which, in addition to well-known performance figures, infers TCP connection status from traces. By passively listening traffic on a bidirectional link, Tstat correlates incoming and outgoing packets. Sophisticated statistics, obtained through data correlation between incoming and outgoing traffic, give reliable estimates of the network performance also from the user perspective. Tstat computes over 80 different performance statistics at both the IP and TCP layers, allowing a good insight in the network performance; the derived statistics include parameters such as the congestion window size, the number of out-of-sequence segments and duplicated segments, etc. The detailed description of Tstat can be found in [6].

B. Network setup

We consider an operative WLAN installed at Dipartimento di Elettronica of Politecnico di Torino. It consists of an access point (AP) using IEEE 802.11b standard; WEP encryption and MAC address filtering are enabled for security reason. Unless otherwise indicated, the RTS/CTS mechanism is disabled. The AP is connected to a Linux based NAT router, which allows wireless terminals to access the Internet.

The AP is located in the middle of the corridor (about 35m long) of our department, while several laptops with wireless access facilities are distributed in the offices along the corridor. Figure 1 shows a map of the corridor, the offices and the AP and laptops position. It also reports the logical

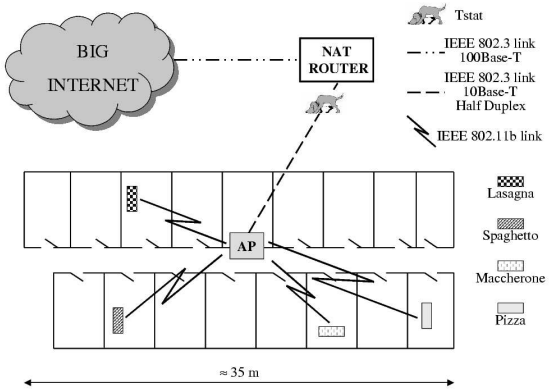

Fig. 1. The map of the physical test setup, and network configuration.

topology, together with links capacity. As it can be seen, the link between the AP and the NAT router is a 10Base-T half-duplex IEEE 802.3 link. Indeed, during the first phase of this experiment, we discovered that using a faster setup negatively affected the performance of the wireless network. This was due to congestion that arises at the AP queue when forwarding packets from the wired LAN to the (slower) wireless LAN. Therefore, in order to minimize the effect of droptail packet drops at the AP queue, we forced the wired LAN to the minimum possible speed.

In the experiments reported in this paper we use four different laptops named Lasagna, Maccherone, Pizza and Spaghetto. In order to limit the number of parameters that could possibly influence the performance, all the laptops use the same wireless card and run under the same operating system (OS). In particular, we used Orinoco Silver PCMCIA cards and we opted for Knoppix Version 3.3 [7], a Linux distribution based on Linux kernel 2.4.24 that runs directly from CD. We selected Linux as OS to have full control on the application that can access the network, and to control easily some TCP parameters, such as MTU, TCP window size, and so on. In the tests we report, all parameters were configured using their default values selected by the OS, except where explicitly reported; in particular, we recall that Linux implements TCP SACK version. Table I reports the average SNR as reported by the OS monitoring tool on each of these laptops. Values are averaged over 10 different measurements and are reported here as pure reference, since the quality of the channel could change during the reported experiment due to external factors, e.g., people moving around, or opening/closing the office doors. Moreover, even if the noise measure could be unreliable, we are confident that, since we use the same OS and wireless cards, the values reported are meaningful as indications of the relative channel quality perceived by the laptops. Lasagna, Maccherone and Spaghetto, the closest laptops to the AP, obtain a very good SNR, while Pizza has only about 12dBm as SNR, even if it is only about 15m far from the AP. This is not only due to its distance from the AP but also to the presence of some obstacle like metallic closets which cause electromagnetic effects that negatively impact the quality of

TABLE I

SNR MEASURED BY THE DIFFERENT LAPTOPS USED DURING THE TESTS

Name	Lasagna	Spaghetto	Pizza	Maccherone
SNR [dBm]	31.5	26.4	11.9	29.1

the transmission.

On the wired LAN, the NAT router (based on Linux kernel 2.4.20) acts also as master (or slave, depending on the setup), and it runs Tstat to analyze the traffic. The CPU load was double checked to be sure not to affect the traffic generation and measurements.

III. SELECTED MEASUREMENTS

In all the selected plots shown in this section except the last one, a point is drawn for each completed data transfer. The x-axis coordinate corresponds to the time when the transfer completes, defined by Tstat as the time instant the last segment containing data from the server-side has been observed, i.e., neglecting the TCP connection tear down procedure which may be delayed by the OS without affecting the user perception of performance. The y-axis coordinate reports the value of the considered performance index computed for each data transfer. Among all the possible measurements Tstat performs, we selected:

- *Transfer time:* defined as the time interval between the first (client) SYN segment and the last (server) segment carrying data.
- *Throughput:* computed by dividing the file size D by its transfer time.
- *Round Trip Time (RTT):* evaluated by Tstat by mimicking the RTT estimation algorithm implemented in TCP on the observed data segments and their corresponding acknowledgment segments (ACKs); maximum, minimum and average RTT are available.
- *Number of retransmitted segments:* counts the number of duplicate segments observed by Tstat. When co-located with the master, this is a direct measurement of retransmitted packets from the server, which may be due to i) a data segment loss, or ii) an ACK loss, or iii) a server TCP retransmission timer firing before the reception of the corresponding ACK.
- *Congestion window:* computed as the difference between the highest data sequence number sent by the server and the highest acknowledgment number received by it. This value, which is actually the number of in-flight segments, is equivalent to the server congestion window when Tstat and the server-side application run on the same host.

A. Downlink traffic

In the first scenario we consider, most of the traffic flows in the downlink direction, from the AP to the wireless stations: each laptop acts as a slave, and the LAN router acts as master. All slaves were instructed by the master to request D = 1MByte of data for $N_t = 100$ trials ($T_{off} = 0$). By controlling the LAN router (i.e., the server during download phase) TCP maximum congestion window (CWND), we consider two

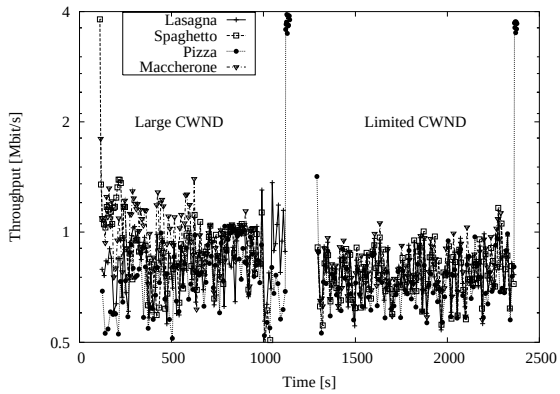

Fig. 2. Throughput versus time. Downlink scenario.

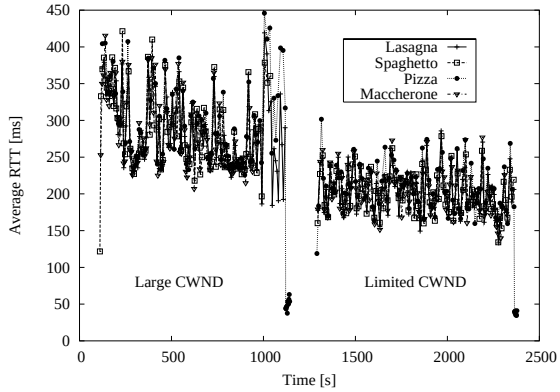

Fig. 3. Average RTT versus time. Downlink scenario.

cases. During the first cycle the server maximum CWND is left to its default value, 64kBytes; instead, it is limited to 32kBytes during the second cycle.

Figure 2 reports the throughput of each connection versus time using a lin-log scale. During the first cycle, which lasts for about the first 1100s and is labeled "Large CWND" in the plot, each laptop downloads 1Mbyte of data for 100 consecutive times. No large difference in the throughput is observed: Maccherone and Spaghetto show the (slightly) largest throughput, while Pizza experiences the worst performance, in accordance with the measured channel quality. By getting smaller throughput, Pizza ends the data download later, and therefore it has the chance to remain the only terminal to access the network, reaching a value of throughput larger than 3.5Mbps during the last part of its download phase.

At about 1200s since the beginning of the experiment, the second download cycle starts with the server maximum CWND limited to 32kBytes (case labeled "Limited CWND" in the plot). Compared with the previous case, the throughput variability reduces. Similar to throughput, even RTT is more variable in the Large CWND case than in the Limited CWND case, as can be observed from Figure 3 and Table II, which reports, for each laptop, the mean and standard deviation of the RTT experienced during 100 downloads. The reason why there is quite a large variance during the Large CWND experiment, especially considering different downloads from the same laptop, is the following. Under large values of maximum CWND,

TABLE II

MEAN AND STANDARD DEVIATION (IN BRACKETS) OF THE THROUGHPUT AND RTT.

Host	Large CWND		Limited CWND	
	Thru [Mbps]	RTT [ms]	Thru [Mbps]	RTT [ms]
Lasagna	0.84 (0.130)	281.2 (48.8)	0.78 (0.114)	201.2 (30.7)
Spaghetto	0.93 (0.171)	286.0 (49.2)	0.77 (0.114)	200.8 (28.4)
Pizza	0.77 (0.141)	289.7 (50.0)	0.72 (0.100)	210.6 (30.4)
Maccherone	1.00 (0.141)	282.6 (44.1)	0.80 (0.114)	202.1 (29.2)

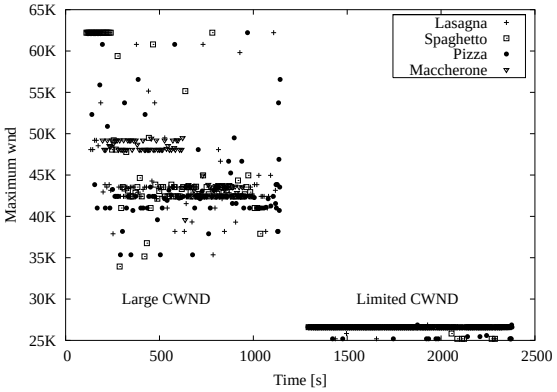

Fig. 4. Maximum server window versus time. Downlink scenario.

TCP fills the AP queue causing segment dropping. Smaller values of maximum CWND, on the contrary, allow the AP queue to absorb the burst of incoming packets. Moreover, since limited maximum CWND results in a less aggressive TCP behavior, and rarely induces segment losses at the AP queue, an increased fairness among stations and among downloads can be observed. This points directly to TCP congestion control as major cause of unfairness. Notice also that when Pizza is the only active receiver, i.e., at the end of a cycle, the experienced RTT is reduced to about 50ms, i.e., no queue builds up at the AP.

Figure 4 reports the maximum server window observed for each download. By setting a larger maximum server CWND, some packet drops occur at the AP queue, which becomes the network bottleneck. TCP reacts to the drop/congestion events by limiting the maximum CWND. On the contrary, limiting a priori the maximum CWND value results in almost all TCP connections to reach the maximum CWND, without overloading the AP queue. Clearly, this can negatively affect throughput; however, if the CWND setting is not too tight, the effect on TCP throughput is marginal, as TCP can still fill the bandwidth/delay product pipe.

B. Uplink traffic

In the second scenario we consider, most of the traffic flows in the uplink direction from the wireless stations to the AP: a server resides in each laptop and a client for each server resides on the LAN router. Clients request $D = 1$MByte of data for $N_t = 100$ trials with $T_{off} = 0$. In this scenario, the MAC algorithm plays a more relevant role than in the downlink scenario, given that there are four transmitters competing for the channel.

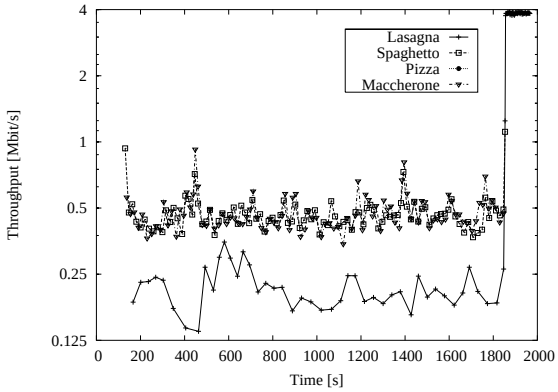

Fig. 5. Throughput versus time. Uplink scenario with simultaneous beginnings of the download phases.

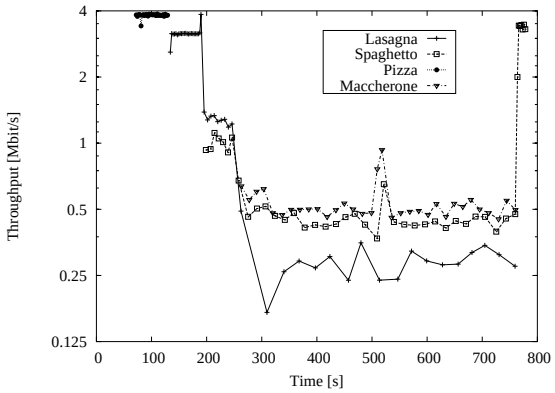

Fig. 6. Throughput versus time. Uplink scenario with delayed beginnings of the download phases.

Figure 5 reports throughput versus time. Probably due to its unfavorable SNR, Pizza is severely penalized and cannot terminate *any* data transfer. Lasagna throughput is not as high as those of the other two laptops, but, at least, it can terminate its data transfers (as expected, it can highly improve its throughput as soon as Maccherone and Spaghetto conclude their data transfers).

In order to further investigate the service discontinuity occurring to Pizza, we let the laptops enter the WLAN in the reverse order of their perceived performance. Figure 6 shows that, when Pizza is alone, its throughput is high. As soon as Lasagna starts competing for the bandwidth, Pizza cannot complete any transfer. However, the presence of Pizza attempting to transfer data interferes with Lasagna performance, whose throughput is smaller than the 4Mbps received by Pizza when it was alone. The competition for the bandwidth between Pizza and Lasagna results in a systematic exclusion of Pizza. On the contrary, when a third station, Spaghetto, enters into play, the bandwidth is quite equally shared between Lasagna and Spaghetto, Pizza still starves and disturbs (the total throughput is far below the initial 4Mbps). Finally, when all laptops are active, unfairness toward Lasagna can be clearly identified, in the same way as in the previous experiment with simultaneous beginnings of data transfers. The fact that, when alone, Pizza behaves normally, whether it is completely starved

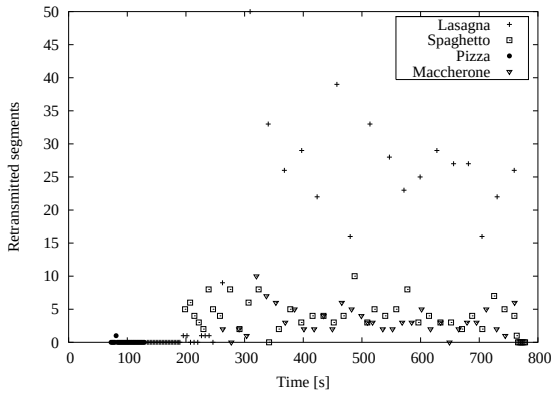

Fig. 7. Number of TCP retransmitted segments versus time. Uplink scenario with delayed beginnings of the download phases.

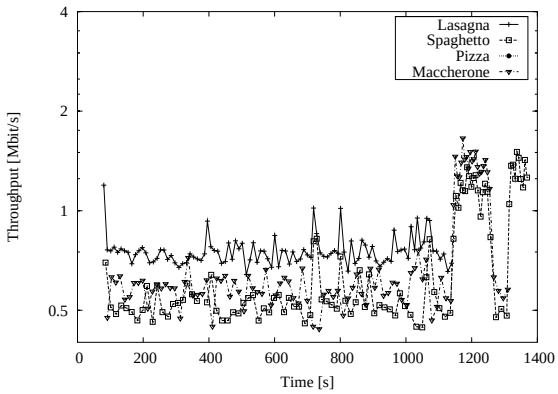

Fig. 9. Throughput versus time. Uplink scenario with RTS/CTS mechanism.

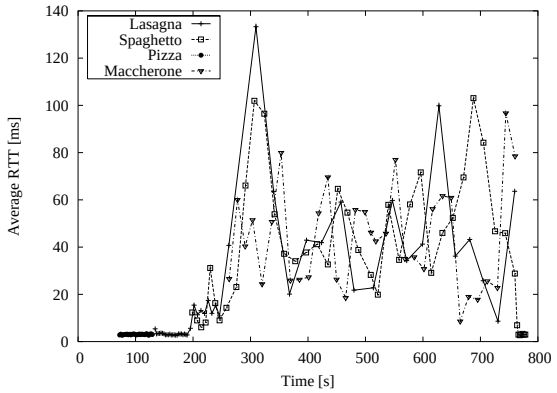

Fig. 8. Average RTT versus time. Uplink scenario with delayed beginnings of the download phases.

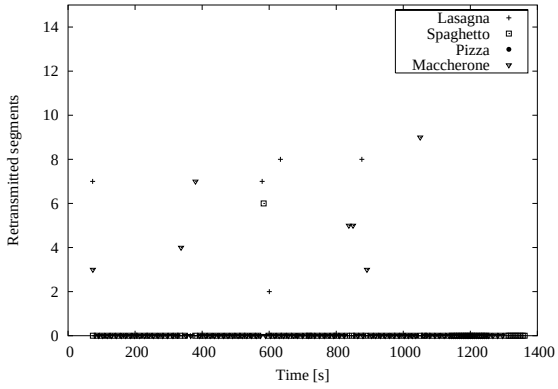

Fig. 10. Number of retransmitted segments versus time. Uplink scenario with RTS/CTS mechanism.

when multiple stations access the channel, points directly to the MAC algorithm, rather than to physical layer problems. A confirmation of this comes also from Figure 7, which reports the number of retransmitted segments in the experiment with delayed beginnings of the download phases. As far as Pizza is alone, no segments are lost, this meaning that the unfavorable channel conditions are not that bad: the link layer retransmission mechanism can recover all data losses on the channel such that no retransmissions are needed at the TCP layer. The large number of retransmitted segments for Lasagna confirms also that Lasagna performance is worse than that of Spaghetto and Maccherone. Finally, as can be observed in Figure 8, the RTT exhibits large variance: this is a consequence of random contentions for channel access which are frequent in the uplink scenario where almost all the traffic flows from laptops to the AP. The value of the RTT in the uplink scenario is much smaller than the one observed in the case of downlink traffic; indeed, when most of the traffic is downlink, queueing delay at the AP makes the RTT significantly increase.

We now wonder whether the RTS/CTS mechanism can eliminate the service discontinuity that Pizza undergoes. We therefore enable the RTS/CTS mechanism with threshold equal to 400Bytes and repeat the experiment discussed above with simultaneous beginning of downloads from all the stations. As can be observed in Figure 9, RTS/CTS mechanism does not succeed in letting Pizza enter into play. Again, Pizza cannot complete any file transfer. On the contrary, compared to the case with no RTS/CTS (see Figure 5), fairness between Spaghetto, Lasagna and Maccherone slightly increases; Lasagna being this time favored. Interestingly, it can be noted that, despite the larger overhead introduced by RTS/CTS, the total throughput, as well as the throughput per connection, increases. As shown in Figure 10, this is due to the beneficial effect of the mechanism in reducing TCP segment loss probability, which is much smaller when RTS/CTS is enabled.

As a conclusion, the introduction of RTS/CTS mechanism increases fairness between TCP connections sharing the same radio resource; moreover, by reducing losses, the mechanism improves the total throughput. However, severe unfairness, such as the one Pizza undergoes, cannot be eliminated by the RTS/CTS mechanism.

In order to try and overcome this unfairness, we investigate the impact of the wireless card rate on the performance. While in previous situations each card rate was simply left to the automatic setting, we now set the transmission rate to specific values. The considered scenario is the one with uplink traffic, no RTS/CTS mechanism, and simultaneous beginnings of the download phases.

In Figure 11, once again reporting throughput as a function of time, all wireless cards are forced to work at the same rate, that is equal to 11Mbps for plots on the left and to

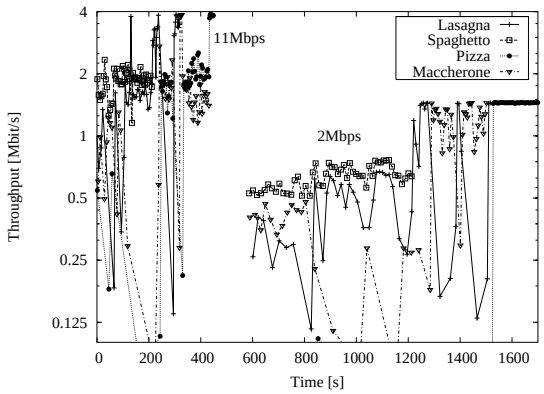

Fig. 11. Throughput versus time of TCP connection. Uplink scenario with all wireless cards rates set equal to 11Mbps (left) and 2Mbps (right).

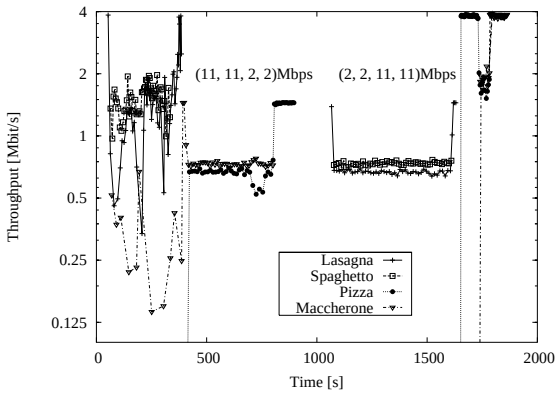

Fig. 12. Throughput versus time of TCP connection. Uplink scenario with wireless cards rates set equal to 11Mbps or 2Mbps; laptops with the same rate are Pizza and Maccherone, Spaghetto and Lasagna.

2Mbps for plots on the right. As could be expected, throughput is significantly higher when the transmission rate is set to 11Mbps; the whole experiment requests about 400s at 11Mbps, while more than 1000s are needed for the 2Mbps case. In both cases, occasionally Pizza can terminate a data transfer with very low throughput before the other laptops terminate their transfers. The drawback of higher throughput at 11Mbps is higher variance and a significant unfairness (Spaghetto performance are almost always better than those of other laptops).

A different setting was chosen for results in Figure 12. The wireless card rate is manually set such that laptops transmission rates are coupled: Pizza and Maccherone set the rate to 11Mbps (2Mbps), Spaghetto and Lasagna to 2Mbps (11Mbps). The main message from Figure 12 is that changing card bit rate selectively does not improve Pizza performance. This confirms the assumption of MAC rather than physical layer problems. On the left side of the Figure, we can see that Pizza is able to start performing data transfers only after Lasagna and Spaghetto finish their downloads (Maccherone succeeds before this happens but gets poor performance). We still observe large variance of throughput values in the time window where Lasagna and Spaghetto, running at 11Mbps, perform their data transfers. In this phase, TCP congestion control kicks in even if no real congestion is present; this is no more true when Lasagna and Spaghetto leave the play. The opposite selective bit rate setting, on the right side of the Figure, further amplifies the problems Pizza and Maccherone face in accessing the channel. Indeed, now, also Maccherone performs its first transfer when Lasagna and Spaghetto conclude all theirs. Finally, differently from the previous selective setting, Lasagna and Spaghetto receive a constant service rate, due to their low card bit rate.

From these results we conclude that the use of high bit rates has a twofold effect. From the one hand, it increases throughput; from the other hand, it increases unfairness. However, since even low rates do not guarantee fairness to a satisfactory degree, the choice of higher bit rates should be more convenient, especially in real traffic scenarios, with the typical intermittent behavior of users.

IV. Conclusions and future work

In this paper, we presented a promising methodology for performance evaluation of WiFi networks fed by TCP traffic. By running a set of experiments on an operative network, we derived some interesting guidelines for the setting of TCP and link layer parameters. We showed that fairness issues may arise, especially for uplink traffic scenarios. Such issues can be so severe that they cannot be eliminated by using any of the standard options, such as RTS/CTS mechanism, and transmission rate selection. The greedy and bursty congestion control of TCP exacerbates fairness issues, and degrades performance unless explicit solutions are considered, i.e., imposing maximum congestion window.

As future work, we plan to extend the measurement campaign considering more realistic traffic scenarios, in particular focusing on variable size flows which include short-lived flows and mixes of TCP and UDP traffic.

References

[1] M.Mellia, R.Lo Cigno, F.Neri, "Measuring IP and TCP behavior on edge nodes with Tstat", *Computer Networks, 47*, Vol. 1, pages. 1-21, Jan 2005.

[2] G. Xylomenos, G. C. Polyzos, P. Mahonen, M. Saaranen, "TCP Performance Issues over Wireless Links", *IEEE Communications Magazine*, Volume: 39, Issue: 4 , Pages: 52-58, April 2001.

[3] G. Anastasi, E. Borgia, M. Conti, E. Gregori, "IEEE 802.11 ad hoc networks: performance measurements," *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 19-22 May 2003, pages:758 - 763.

[4] K. Benekos, N. Pogkas, G. Kalivas, G. Papadopoulos, A. Tzes, "TCP performance measurements in IEEE 802.11b-based wireless LANs," *12th IEEE Mediterranean Electrotechnical Conference (MELECON)*, 12-15 May 2004, vol.: 2, pages: 575 - 578.

[5] M. Franceschinis, "GenTraGen: Generic Traffi c Generator", http://www.tlc-networks.polito.it/software.html, 2005.

[6] M. Mellia, A. Carpani, R. Lo Cigno, "Tstat web page", http://tstat.tlc.polito.it/, 2004.

[7] K. Knopper, "Knoppix home page", http://www.knopper.net/knoppix/index-en.html, 2003.