TCP Fairness Issues in IEEE 802.11 Networks: Problem Analysis and Solutions Based on Rate Control

Nicola Blefari-Melazzi, Andrea Detti, Ibrahim Habib, Alessandro Ordine, and Stefano Salsano

Abstract—**In this paper, we study the problem of maintaining fairness for TCP connections in wireless local area networks (WLANs) based upon the IEEE 802.11 standard. Current implementations of 802.11 use the so-called Distributed Coordination Function (DCF), which provides similar medium access priority to all stations. Although this mode of operation ensures fair access to the medium at the MAC level, it does not provide any provisions for ensuring fairness among the TCP connections. TCP unfairness may result in significant degradation of performance leading to users perceiving unsatisfactory quality of service. We propose and analyze two solutions that are capable of enabling TCP fairness with minimal additional complexity. The proposed solutions are based on utilizing a rate-control mechanism in two modes: static or adaptive. They do not require modifying existing standards at the MAC or network layers. Hence, they are fully compatible with existing devices. Our performance analysis results prove the efficaciousness of our proposed solutions in achieving TCP fairness compared to existing approaches. We have, also, implemented the proposed solutions in an ad-hoc experimental test-bed, and performed measurements to demonstrate the validity of our approach and results.**

Index Terms—**WirelessLAN, TCP fairness, rate limiter.**

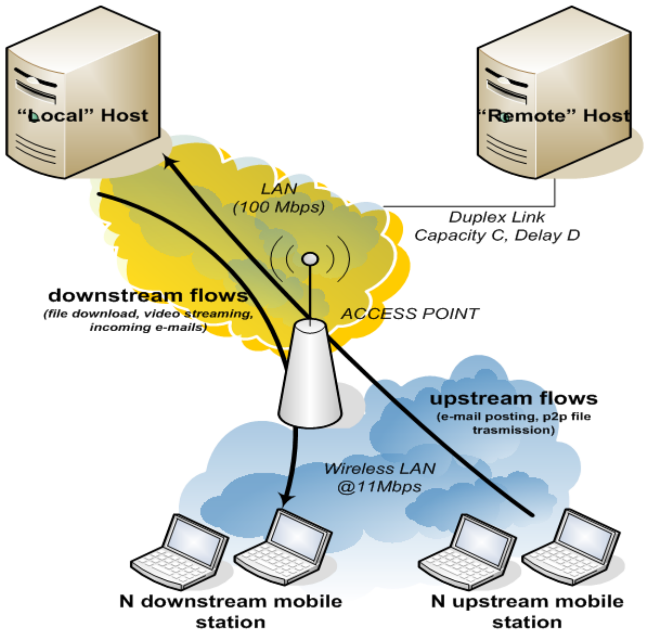
"Local" Host
"Remote" Host
LAN (100 Mbps)
Duplex Link Capacity C, Delay D
downstream flows
(file download, video streaming, incoming e-mails)
ACCESS POINT
upstream flows
(e-mail posting, p2p file trasmission)
Wireless LAN @11Mbps
N downstream mobile station
N upstream mobile station

Fig. 1. Reference simulation scenario.

I. Introduction

WIRELESS LANs based on the IEEE 802.11 standard and working in the so-called "infra-structured" mode provide mobile stations with access to the wired network by means of "Access Points". Such "infra-structured" WLANs are to be distinguished from the "ad-hoc" WLANs, where mobile stations talk to each other without using Access Points.

It is important for infra-structured WLANs, especially when offering public access to the Internet, to maintain fairness among TCP connections competing for access to the shared media of the WLAN. By fairness among multiple TCP connections, we mean that any TCP engine would be capable of starting a connection with negligible delay, as well as achieving and maintaining a reasonable throughput. The latter, of course, depends on other competing TCP connections. Viewed this way, TCP fairness is, then, a mandatory prerequisite for enabling a satisfactory quality service for upper layer applications. However, we also have to specify that we are not requesting a "perfect" fairness, i.e., a perfectly balanced sharing of resources among all TCP connections (which can be seen as a "second order" objective). Rather, our main aim is to avoid the scenario of "*critical unfairness*" that is characterized by complete starvation of some TCP connections or, even, the inability of some TCP connections to start altogether.

This so called *critical unfairness* can arise in two cases: 1) interaction between upstream and downstream TCP connections, or 2) interaction between a set of upstream TCP connections. TCP connections are labeled as "downstream" or "upstream" (see Fig. 1), depending upon the direction of traffic flow. Downstream is used to describe traffic flowing from the wired network towards the mobile station (e.g., file downloads from a web server, video streaming, incoming e-mails), whereas Upstream is used to refer to traffic flowing from mobile stations to the wired network (e.g., e-mail posting, peer-to-peer file transmission, etc.).

In the following we introduce the two critical unfairness

Manuscript received June 4, 2005; revised February 4, 2006 and November 29, 2006; accepted January 18, 2007. The associate editor coordinating the review of this paper and approving it for publication was X. Shen.

N. Blefari-Melazzi, A. Detti, A. Ordine, and S. Salsano are with the Dipartimento di Ingegneria Elettronica, University of Rome "Tor Vergata," Rome, 00133 Italy (e-mail: {blefari, andrea.detti, alessandro.ordine, stefano.salsano}@uniroma2.it).

I. Habib is with the Department of Electrical Engineering, City University of New York (e-mail: habib@ccny.cuny.edu).

Digital Object Identifier 10.1109/TWC.2007.05425.

cases that will be thoroughly analyzed in the next section. In the first case, downstream TCP connections are penalized with respect to upstream ones. This is explained as follows: packets belonging to multiple downstream TCP connections are buffered inside the Access Point wireless interface. Note that the Access Point does not enjoy a privileged access to WLAN capacity, with respect to user terminals. Hence, a single station transmitting upstream packets will get the same priority as that of the Access Point which needs to transmit downstream packets heading towards many stations. Thus, downstream TCP connections suffer because of the arising congestion and corresponding packet losses happening in the download buffer at the Access Point. These losses in conjunction with TCP congestion control mechanism cause the starvation of downstream connections. This is defined as "critically" unfair.

The second case arises from the interaction of multiple TCP connections in the upstream direction. In this case, the Access Point wireless interface has to transmit TCP ACK packets traveling downstream towards stations in the WLAN. Also in this case we have a bottleneck, because the Access Point can not access the medium with a priority higher than other stations. Hence, the Access Point buffer will be congested, leading to severe loss of TCP ACK packets. Due to the cumulative nature of TCP ACKs, few connections will be able to "survive" and open their window, while the majority of connections will get starved. Note that this situation is not specific of our scenario; it can happen in whatever environment characterized by heavy losses of ACK packets. This case is also another example of "critical unfairness" which will be explained in more details in the next section.

It is worth-mentioning that the 802.11 standard also includes a different access control mechanism, called Point Coordination Function (PCF). An extension of the basic 802.11, namely the draft standard 802.11e, provides further mechanisms to control the allocation of the WLAN resources. Both the PCF and the 802.11e could be used to improve the fairness perceived at the application level. Unfortunately, the current status is that the large majority of existing WLAN cards and devices support neither the PCF functionality nor the 802.11e. Considering the lack of deployment of PCF and 802.11e, we focus on strategies to achieve TCP fairness by using the widely deployed DCF mechanism. Moreover, we focus our attention only on techniques that can be implemented within the Access Point (or in a nearby router), *without requiring changes in the 802.11 standard, nor any enhancement to mobile stations.*

As for the organization of the paper, in Section II we briefly summarize related works and introduce some basic assumptions. In Section III we analyze the "basic" system model, i.e., an infra-structured WLAN, without rate control mechanisms. In Section IV, we introduce our solutions to face the so called critical unfairness, based on rate control mechanisms. The static approach is detailed in Section V, together with the related performance evaluation; similarly, we present the adaptive rate control in Section VI, together with the related performance evaluation. In Section VII we discuss conclusions and future work.

Finally, we point out that the interested reader may find in [6] some details and additional results that could not find space in this paper, including: i) the assessment of the impact of different TCP version on our solution (in Appendix I); ii) the performance evaluation of our mechanisms for different values of the Round Trip Time, showing that our solution works independently from this parameter (in Appendix II); iv) a description of our test-bed, which shows how our solution perfectly works in a real environment (in Appendix IV); v) the performance evaluation of our system when loaded with short-lived TCP sources and with a mix of short-lived and greedy sources, two traffic scenarios that extend the one considered throughout this paper (in Appendix V).

II. Related Work and Basic Assumptions

The problem of WLAN fairness at MAC level has been analyzed in several papers, (e.g., [1], [2]); however, MAC level fairness is not enough. Unfairness among TCP connections within a single WLAN was analyzed in [3]; in which the authors propose an elegant solution that addresses not only the so-called critical unfairness (i.e., it avoids connection starvations), but also a finer "per connection" fairness. However, the approach proposed in [3] has some drawbacks in terms of implementation complexity, dependence on connection Round Trip Times, and need to parse the TCP header, which cannot be accessed in case IPSec is used (see Section IV). Unfairness among TCP and UDP flows in more complex topologies (i.e., with multiple WLANs) has been preliminary discussed in [4].

In this paper, we propose solutions aiming at avoiding critical unfairness (i.e., starvation) and at enforcing a fair sharing of radio bandwidth between the Access Point and the mobile stations. Our solution works on the aggregate TCP flows crossing the Access Point. Consequently, we do not provision a perfect "per connection" fairness. However, our solution is simple to implement, robust against variable Round Trip Times, and does not require parsing of TCP headers. Hence, it is also compatible with IPSec.

Our approach is based upon utilizing a rate-limiter to control the overall uplink rate, with the aim of avoiding the critical unfairness. The rate-limiter is implemented via a Token Bucket Filter (TBF) [5]. The TBF is characterized by two parameters: 1) the rate of generating tokens into the bucket (R), and 2) the capacity of the bucket (bucket depth, B_{bucket}). The TBF generates tokens at rate R and puts them in the bucket. The rate limiter forwards arriving uplink packets only if there are enough tokens available in the bucket, otherwise uplink packets are dropped. The TBF forces packets' loss when the uplink aggregate rate is higher than the TBF rate R and the bucket does not contain enough tokens. The TCP congestion control mechanisms, that are automatically enabled when losses are detected, reduce the transmission windows and consequently the number of transmitted packets. Thus, by setting the TBF rate R and bucket depth B_{bucket}, it is possible to suitably control the overall uplink rate.

We also assume that the parameter R can be either statically configured or it can be dynamically and adaptively varied as a function of an estimation of the attainable downstream throughput (this choice will be better motivated later on).

We will refer to these two alternatives as static rate control and dynamic rate control, respectively. We will show that these solutions are indeed very simple to implement and that the adaptive rate control is very effective in avoiding the starvation of TCP connections and resource wasting. In addition, our approach can provide the operator of the WLAN with a tool to controlling the sharing of WLAN resources between upstream and downstream applications.

III. System Model and Performance Measures

The simulation model is shown in Fig. 1[1]. A number of wireless stations are connected to an Access Point and exchange information with a host in the high-speed fixed network (this host being labeled as "*wired host*"). In particular, we consider N_{dn} wireless stations downloading information from a wired host and N_{up} wireless stations uploading information to a wired host.

As shown in Fig. 1, in our simulation environment the wired host can be connected to the Access Point via a Fast Ethernet (100 Mbit/s full duplex) LAN link, or via a generic duplex link with capacity C and one-way propagation delay D. The former represents the case in which the Access Point and the wired host are in the same Local Area Network ("local wired host"). The latter represents the case in which the wired host is remotely located somewhere in the Internet ("remote wired host"). We can set the Round Trip Time (RTT) between the wireless stations and the remote wired host to arbitrary values by choosing a proper value of D. In the same way, we can emulate a bottleneck in the Internet connection toward the remote wired host by properly setting the capacity C. Simulations have been carried out by using the NS-2 simulator package (version 2.1b9a) [7]. Within this environment, we suitably defined the simulation scenario and wrote the necessary additional code; the most important simulation parameters that we adopted in this work are: IP packet size: 1500 bytes. Maximum TCP congestion window: 43 packets (64 kbytes). TCP version: Reno [9]. The latter choice stems from the fact that this is the version currently installed in commercial Microsoft Windows based PCs as well as in typical Linux distribution. However, we also tested TCP NewReno [10] and SACK [11], to verify that the phenomena under study are not specifically tied to the selected TCP version (see Appendix I in [6]). As regards the traffic loading the system, we assume two different traffic source models: i) greedy sources, that is TCP sources that have always information to transmit - this model is denoted in the literature also as "infinite file transfer" model; ii) short-lived TCP sources, modeling the download or the upload of a small amount of data. The short-lived traffic scenario is described and analyzed in the Appendix V in [6]; the greedy sources traffic scenario is the main one, described and analyzed through the paper.

As for the downlink buffer, we will present simulations in which we vary its size, B, to analyze the impact of this critical parameter (e.g., B=50, B=100, B=300 packets). When it is not otherwise specified, the downlink buffer size is B=100 packets, which, according to [3], is a typical value for commercial equipments.

For each connection, we evaluated the throughput. We denote by throughput the bit rate transported by the layer below IP, comprehensive of all upper layers overheads (including IP and TCP) and of the overhead resulting from TCP ACKs flowing in the reverse direction[2]. Also, we denote by *upstream* the direction of an asymmetric TCP connection whose greater part of data flows from a mobile station to the wired network and by *uplink* the physical direction of packets going from a mobile station to the wired network. Similar definitions apply to *downstream* and *downlink*. This implies, for instance, that both uplink and downlink packets flow within an upstream TCP connection. The figures of merit that we consider are: the total upstream throughput R_{up_tot} (i.e., the sum of the throughputs of upstream TCP connections), the total downstream throughput R_{dn_tot} (i.e., the sum of the throughputs of downstream TCP connections), and the total throughput R_{tot} (the sum of upstream and downstream throughputs).

Unfairness between upstream and downstream TCP connections occurs when $R_{dn\_tot} << R_{up\_tot}$, assuming that both upstream and downstream TCP connections are active and "greedy".

To analyze unfairness among flows heading in the same direction, for instance in the upstream one, we evaluate the ratio between the standard deviation (σ_{up}) and the mean value ($R_{up} = R_{up\_tot}/N$) of the throughput of upstream connections. If the ratio σ_{up}/R_{up} is zero, then we have perfect fairness; otherwise unfairness increases as this ratio increases. The same applies for the downstream case, considering the downstream ratio σ_{dn}/R_{dn} (with $R_{dn} = R_{dn\_tot}/N$). In the following, this ratio will be called unfairness index.

A. Numerical Results

We analyze an infra-structured WLAN without rate control mechanisms and we assume that wired hosts are locally connected to the Access Point (scenario "local wired host"). We also assume that $N_{dn} = N_{up} = N$ i.e., that the number of upstream connections is equal to the downstream ones. Simulation experiments lasted 600 seconds of simulated time; the throughput was measured during the last 200 seconds of each experiment, to avoid transient effects.

Fig. 2 shows the upstream throughput, the downstream throughput and the total throughput (i.e., the sum of these two components) in the WLAN as a function of N ($N = N_{dn} = N_{up}$) and for B=100 packets. For N=1, i.e., when there is only one upstream connection and one downstream connection, the overall bandwidth is fairly shared. The throughput of downstream connections drastically decreases as N increases and is almost equal to zero for N=4. Thus, downstream connections do not succeed in perceiving their "right" bandwidth share, even with a moderate number of upstream connections. The total throughput slightly increases with N, as an indirect consequence of the increase in packets' loss in the downlink

[1]Throughout this paper, we will not present the 95% confidence intervals of simulation results, in order to improve the neatness of the figures. However, such intervals are always less than 5%.

[2]This is a better indication on how the WLAN resources are being used, rather than the TCP goodput, which does not take into account the IP headers and the TCP ACKs.

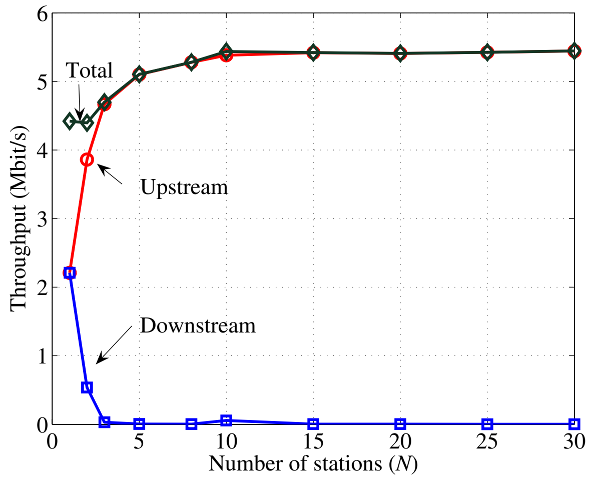

Fig. 2. Upstream, downstream and total throughput ("local wired host" scenario) without rate control mechanisms.

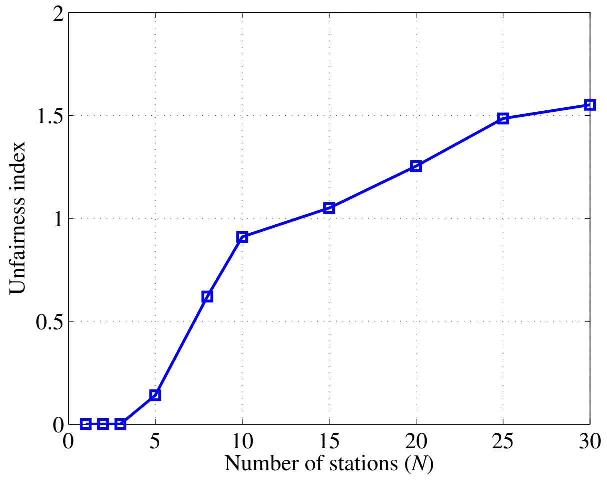

Fig. 3. Ratio σ_{up}/R_{up} for upstream connections ("local wired host scenario").

buffer [3].

If we look at individual upstream connections, and we measure the throughput that each connection perceives, in the same setting of Fig. 2 (as done in [6]), we find out that there is no fairness at all, as soon as the number of flows is greater than 5. In fact, some flows do not even succeed to starting transmission, while other flows seize all the WLAN resources(for N=20 only 8 upstream flows actually use WLAN resources). This is confirmed by the values of the ratio σ_{up}/R$_{up}$, plotted in Fig. 3, which sharply increase for N>5.

B. *Understanding the results*

This section analyzes the results shown above. We state that the main cause of starvation, and unfairness, is the packet loss occurring in the downlink buffer. The causes of packet loss are first highlighted, then it is explained why such loss results in unfairness, and even starvation, of some connections.

The packet loss in the downlink buffer may attain large values because of the DCF access mechanisms whose task is to, fairly, share the available capacity among all active entities, mobile stations, and Access Point alike. Since the Access Point does not enjoy a privileged access to WLAN capacity with respect to users' terminals and since it has to handle more traffic with respect to a single station; it is more likely that its downlink buffer becomes congested, with respect to the buffering resources of mobile stations. We now investigate why this loss leads to TCP starvation of some connections.

We start by looking at *downstream connections*. For such connections, a packet loss in the downlink buffer means a TCP segment loss; TCP segment losses trigger congestion control mechanisms which, in turn, cause a decrease of the TCP throughput. In addition, both at the beginning of a connection and after the occurrence of several segment losses, the TCP congestion window is small in order to prevent the use of fast retransmit mechanisms. Hence, most of the losses are recovered by means of the Retransmission Time Out (RTO) mechanism. Since the RTO doubles after each consecutive loss (and consecutive losses are likely in the above conditions), downstream connections experience long idle period, and even throughput starvation. To support this hypothesis, we evaluated by simulations the packet loss probability in the downlink buffer of the Access Point as a function of N, and for different values of the buffer size (Fig. 4). The main results are the following. With B=100 packets, when N=1, the downlink buffer at the Access Point is large enough to avoid loss; the downstream throughput is not starved. When N increases, the downlink buffer occupation increases as well, and even for small values of N the loss probability is great enough to starve all downstream connections (e.g., 20% loss for N=3). It is clear that increasing the buffer size allows the handling, in a fair way, of a larger number of sources. However, as the number of sources increases, there is always a point beyond which the loss rate starts to increase, N=6 for B=300, N= 10 for B=500 and N=20 for B=1000 (correspondingly, the total downlink throughput will start to decrease). Thus, the loss rate becomes greater than zero when the number of stations is greater than a threshold which increases with B and then it tends to an asymptotic value.

Let us now turn our attention to *upstream connections*. In this case, a packet loss at the downlink buffer means the loss of a TCP ACK. For large values of such loss probability several consecutive ACKs of the same connection may be lost (e.g., in Fig. 4 we see that the loss probability is greater than 60 % when N=10). The impairments caused by consecutive ACK losses worsen as the TCP congestion window decreases [8]. For instance, assuming ideal conditions, with ACK losses being the only cause of performance degradations, and assuming a congestion window equal to W packets, the sender will find itself in the Retransmission Time Out state, and thus reduce its throughput, only if W ACKs are lost. As

[3]When losses are high, more TCP segments than TCP ACKs are transmitted in the WLAN. If a TCP segment is lost, no TCP ACK is transmitted. If a TCP ACK is lost, it means that a TCP segment has been transmitted while the corresponding TCP ACK is not transmitted. Consequently, the total throughput will increase, since the shorter TCP ACKs (40 bytes) have a proportionally larger overhead as compared to TCP segments (1500 bytes). More details about this phenomenon can be found in [6].

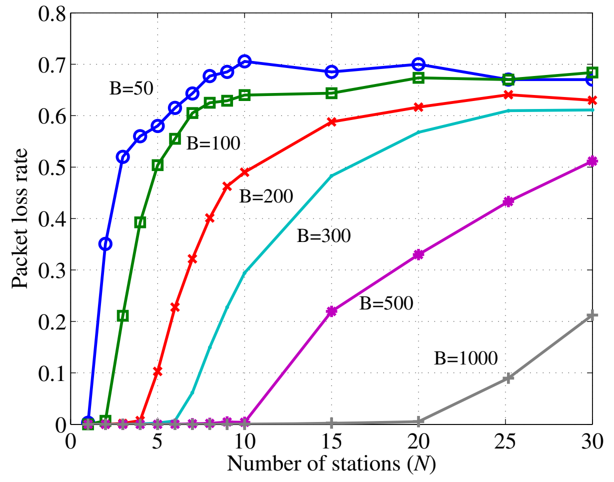

Fig. 4. Packet loss rate in the downlink Access Point buffer ("local wired host" scenario).

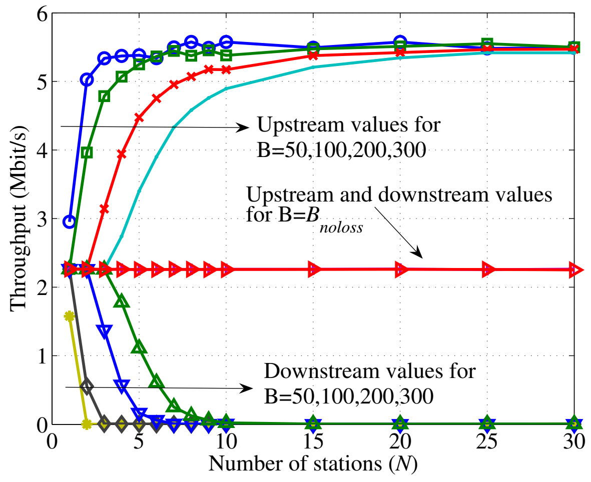

Fig. 5. Total upstream and total downstream throughput.

a consequence, the greater W, the rarer are RTO events. If we consider that the TCP congestion window increases when ACK segments are received, the probability of RTO events is maximized at the start of the connection. On the contrary, these events are always less likely to occur as the congestion window increases, and disappear once the window gets higher than a critical threshold (e.g., five packets). This chain of events is the cause of the fact that some flows do not even succeed to starting transmission, while other flows seize all the WLAN resources. It is worth noting that upstream connections experience starvation for larger values of loss probabilities, as compared to downstream connections. For instance, in our scenario with B=100, the downstream starvation occurs when the loss probability is greater than 20% (and N=3), whereas upstream connections suffer this full service outage for loss probabilities greater than 50% (and N=10).

As a further corroboration of our interpretation, we present more simulation results obtained by varying the downlink buffer size, B. Fig. 5 shows the total upstream throughput and the total downstream throughput as function of N for values of the buffer size ranging from 50 packets to a buffer size large enough to completely avoid loss phenomena, denoted by B_{noloss}. In our scenario, B_{noloss}=2$\cdot N \cdot$CW_max, where CW_max is the TCP maximum congestion window size (expressed in packets of 1500 bytes, following NS-2 conventions). It is worth noting that for any buffer size, increasing the number of connections always leads to starvation conditions. For instance, for a buffer of 50 packets, starvation of downstream connections occurs at N as small as two, whereas for a buffer of 300 packets, the critical threshold of N is seven.

We observe that, with a downlink buffer of size B_{noloss}, all unfairness issues are automatically resolved. In fact, due to the lossless property, the congestion window of all TCP flows can reach its maximum value CW_max (this value being always expressed in packets of 1500 bytes) and all TCP connections get the same amount of bandwidth, in the steady state[4]. This is shown in [6], where we evaluate the throughput of individual connections.

This said, it would seem that, from the starvation problem point of view, the most sensible solution is to choose a size of the downlink buffer equal to B_{noloss}. However, increasing the size of the downlink buffer has the obvious disadvantage of increasing the queuing delay. For example, to support 8 upstream and 8 downstream connections without losses, we need a buffer size in the order of 600 packets. If we consider that: i) half of the buffer will contain TCP segments (1500 bytes); ii) the remaining half will be filled by TCP ACKs (40 bytes), iii) the downlink throughput is in the order of 2.5 Mbit/s, then we can evaluate the queuing delay as being in the order of 300*1500*8/2.5e6 + 300*40*8/2.5e6 = 1.47 s. Hence, TCP connections will experience a rather large Round Trip Time (RTT). In turn, increasing RTTs impair the throughput of short-lived TCP connections. According to these considerations, the buffer size should be set considering the trade-off between maximizing throughput for long-lived TCP connection (large buffer) and minimizing RTT for short-lived TCP connection (short buffer). For the time being, we will follow our analysis by setting the downlink buffer size to a value of 100 packets.

In the next Section, we propose our solution to the critical unfairness problem.

IV. Limiter Based Rate Control

As discussed in the previous Section, we could solve fairness impairments by setting the size of the downlink buffer of the Access Point to B_{noloss}. However, this approach has its disadvantages, some of which have been outlined above. More importantly, it is certainly not easy to influence manufacturers as to make them deploy Access Points with a minimum given buffer size, let alone the issue of all the devices already installed.

An alternative solution, proposed in [3] aims at controlling upstream and downstream rates so that no loss occurs in the downlink buffer. This is done by suitably modifying the window size advertised in TCP packets (such modification

[4]This is true under the assumption that the RTT is the same for all involved connections; this condition tends to be verified as much as the major cause of end-to-end delay is due to the downlink queue of the Access Point.

happening in the Access Point). In this case, fairness conditions are achieved, as discussed in Section III-B [5].

However, we argue that, from an implementation point of view, this solution adds complexity since the device implementing this solution (for example the Access Point itself) must operate on a packet-by-packet basis and parse TCP headers, in order to modify the receiver advertised window. Moreover, it is also necessary to estimate, in real-time, the number of TCP flows crossing such device. This is by no means a trivial operation. The number of TCP connections can change very rapidly, as many TCP connections can be very short-lived. In addition, there can be open TCP connections that are long lived, but which have a minimal activity (for example a telnet connection). These should not be counted among the "greedy" connections competing for the WLAN access bandwidth.

Our proposal is to use a limiter-based Rate Control. This solution could be implemented within the Access Point or in a suitable router of the access network. Additionally, we require that the proposed mechanism operate transparently with actual WiFi mobile stations, based on the DCF defined in the 802.11 standard. Our approach purposely introduces packet losses that trigger the TCP congestion control mechanisms; fairness conditions are achieved by indirectly controlling the *aggregate* rate of TCP connections. The goal is to enforce a fairer sharing of WLAN capacity between the Access Point and the mobile stations. Our rate-limiter operates on the uplink traffic going to the fixed network, at the IP level. Packets are dropped by the rate limiter with the aim of indirectly controlling the rate of uplink flows via the TCP congestion control. The rate limiter is a token bucket filter characterized by two parameters: 1) the rate of generating tokens into the bucket, R (expressed in Mbit/s), and 2) the bucket size B_{bucket} (expressed in kbytes). The TBF generates tokens at rate R and puts them in the bucket. The rate limiter (and thus the AP) forwards arriving uplink packets only if there are tokens available in the bucket, otherwise uplink packets are dropped. Thus, the token bucket operates only as a dropper, i.e., it does not try to reshape non conforming packets and it does not need to queue packets. This makes its practical implementation very simple.

However, the interaction of the simple TBF described above with TCP can lead to annoying synchronization effects, as it is likely that the TBF drops packets in burst, causing several TCP connections to reduce their sending rate in an almost synchronized way. In order to avoid this effect we propose a modified version of the TBF, called Smoothed TBF (STBF). A Smoothed TBF randomly drops packets when the number of tokens in the bucket is greater than zero, very much like a RED queue randomly drops packets before the queue gets full. In particular, let B_{bucket} be the bucket dimension, let H be the current size of the bucket, let $0 < T_h < 1$ be the threshold level at which the STBF should start dropping packets. The STBF introduces a loss probability $P_{drop}(H)$ for an incoming packet as follows:

$$P_{drop}(H) = \begin{cases} 0 & H \geq Th \cdot B_{bucket} \\ \frac{Th \cdot B_{bucket} - H}{Th \cdot B_{bucket}} & H < Th \cdot B_{bucket} \end{cases}$$

We verified by means of simulations that the STBF avoids pseudo-periodic loss pattern that are instead observed when using the simple TBF, and that may lead to synchronization among TCP connections.

We also assume that the parameter R can be either statically configured or it can be dynamically and adaptively varied. We will refer to these two alternatives as static rate control and dynamic rate control and we will analyze them in the following Section V and 6, respectively.

V. Static Rate Control

The static rate-limiter could be implemented within the access point or in an external device (e.g., a router) connected to the access point. The latter solution is especially suitable for a short term scenario or for a test-bed. For example, a Linux based router connected to an "off-the-shelf" access point could constitute an interesting test-bed, useful to make practical experiments. Block diagrams of these two solutions are depicted in [6].

We chose the parameters of the Token Bucket filter, i.e., the rate R and the bucket size B_{bucket}, by means of a simulation study having the aim of identifying such parameters so as to avoid starvation and to provide a fair access possibility to all applications, while maximizing the overall throughput. This simulation study is presented in [6]; the final choice is R=2.3 Mbit/s, B_{bucket}= 500 packets of 1500 bytes and $T_h = 0.9$. We stress that we need to run simulations to choose the parameters of our mechanism only in this static case, which is introduced only as a study case; in the adaptive case we will introduce a procedure to evaluate in real time the parameters of our mechanism, avoiding the need of simulations.

A. Numerical results

In order to evaluate the performance of the proposed rate-control mechanism in the static case, we use the same scenario adopted in Section III, and measure throughput and fairness by varying the number of station N, with $N_{dn} = N_{up} = N$. We first address the "local wired host" scenario. We compare the performances of 3 solutions: i) no rate control, ii) the lossless rate control solution proposed in [3], iii) our proposed static rate limiter. This comparison is reported in Fig. 6, which shows the total throughput as a function of N. The lossless rate control of [3] and our static rate limiter attain almost identical performances: the total throughput is almost constant as a function of the number of sources.

The total upstream and the total downstream throughput are reported in Fig. 7. The lossless rate control of [3] achieves a perfect fairness, as the upstream and downstream curves cannot be distinguished. Our rate limiter solution, however, is slightly less effective in terms of fairness when there are few sources (N=2 to 5), since in this case the upstream connections receive a smaller capacity.

[5]We tested the approach proposed in [3] by means of our simulator and we verified that the results in terms of throughput are equal to those shown in Fig. 5, when the buffer size is equal to B_{noloss}.

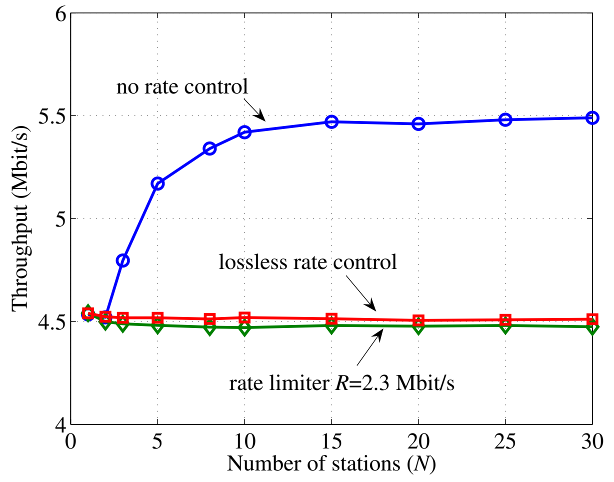

Fig. 6. Average total IP level throughput.

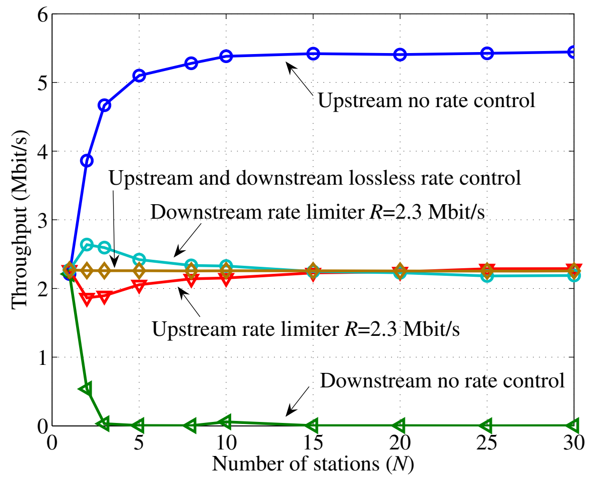

Fig. 7. Upstream and downstream throughput.

We have, also, analyzed throughput performance when N_{dn} is different than N_{up}, verifying that our static rate-limiter is always capable of avoiding the critical unfairness, which is instead evident in the case of no-rate control [6]. In addition, the static rate limiter is capable of enforcing a reasonably good sharing of resources among the upstream and downstream stations close to the ideal target of equal sharing.

As already discussed, our solution does not reach the finer "per connection" fairness (which was not among our goals). However, we have obtained the important result of avoiding the so-called critical unfairness, without the disadvantages of the lossless rate control in terms of complexity.

To conclude the analysis of the rate limiter in the static case, we have performed a simulation study related to the impact of the variable round trip times (RTTs) that TCP connections may experience. With this analysis we have also verified that the proposed rate limiter approach, differently from the lossless rate control proposed in [3], is not affected by limitations related to the RTT. In this simulation study (reported in Appendix II of [6]) we consider the "remote wired host" scenario (see the right part of Fig. 1). TCP connections are terminated on the "remote wired host" and we evaluate the throughput by varying the RTT between the Access Point and the remote wired host itself. This simulation study shows that the performance of the lossless rate control of [3] start to worsen for RTTs greater than 300 ms, whereas the performance of our proposed rate limiter does not depend on RTT at all.

The results presented so far show that the static rate control mechanism enforced by a Token Bucket Filter is effective in avoiding the critical unfairness. In the next Section we highlight the limitations of this static approach and propose an adaptive mechanism.

VI. Adaptive Rate Control

The static mechanism described in the previous Section has two major problems: 1) if downstream connections are not present at all or they are not "greedy", the capacity of upstream connections is un-necessarily limited to the rate R of the Token Bucket Filter, leading to a waste of resources; 2) our static mechanism assumes that the overall capacity C of the WLAN is known and used as an input parameter to the mechanism, since we need to set $R \approx C/2$; however, in general, this capacity is not known since different stations can attach to the Access Point with different physical rates (from 1Mbit/s to 11Mbit/s in 802.11b). To solve these problems we proceed as follows.

Let us denote by C the WLAN capacity and by R the rate of the token bucket filter. If the downstream connections are limited to a rate $R_{down} < C - R$, then the static rate limiter causes a waste of capacity in the order of $C - R - R_{down}$. The idea of the adaptive rate control is to increase the rate of the token bucket filter in these conditions up to R'=$C - R_{down}$ so that no capacity is wasted. When the downstream connections become again greedy, the rate of the token bucket filter is suitably reduced. The proposed mechanism adapts the rate R via discrete steps of amount R_{step} (kbit/s). This adjustment is performed periodically, with an interval of T_p (ms). The choice whether to increase or decrease the token bucket rate is based on a control rule, which takes into account the estimation of uplink and downlink traffic and the information about the packet losses at the AP downlink queue. The uplink and downlink traffic can be estimated outside the AP, and the packet loss information can be extracted from the AP using for example SNMP. Therefore, it is possible to implement the adaptive rate limiter solution in an external device.

Our proposed adaptive mechanism works as follow: each Tp milliseconds we estimate the "instantaneous" throughput (here we use the same definition of throughput given in Section III, i.e., the bit rate transported by the layer below IP, comprehensive of all upper layers overheads, including IP and TCP) crossing the AP in uplink (R_{up}) and in downlink (R_{down}); actually, we use a throughput averaged over a short period, in the order of hundreds of milliseconds. For such estimation, we use an Exponentially Weighted Moving Average (EWMA) algorithm (reported in Appendix III of [6]), which is very simple to implement[6]. We denote by C_{max}

[6]As a comparison, we note that [3] requires to estimate the *number* of TCP connections, which as said above is more difficult to implement.

TABLE I
TIME-SCHEDULE OF THE SIMULATION EXPERIMENTS

Time(sec)	0÷50	50÷100	100÷150	150÷200	200÷250	250÷300	300÷350	350÷400	400÷450	450÷500	500÷550
No. active dw	3	3	0	3	6	6	10	10	10	10	10
No. active up	3	10	10	10	10	6	6	6	0	0	3

TABLE II
ADAPTIVE RATE LIMITER PARAMETERS

Parameters	Values
AP downlink buffer B (packets)	100
TBF bucket size B_{bucket} (bytes)	500*1500
Th	0.9
R_{step} (kbit/s)	200
T_p (ms)	300

the total estimated throughput (i.e., $C_{max} = R_{up} + R_{down}$). At the same time, we monitor the number of losses at the AP downlink buffer (in a real life environment this can be done by SNMP). If no packet losses are observed in the last interval of duration Tp, this means that the downlink buffer is not congested. On the contrary, if there is at least one packet lost this means that the downlink buffer is full. In the former case, we can give more room to the upstream connections (increasing the rate of the Token Bucket Filter), in the latter case we reduce the rate of the Token Bucket Filter to avoid the risk that upstream connections will increase too much their rate, ultimately leading to starvation of TCP connections. The adaptive algorithm, which runs periodically at the time instants T=kT_p, can be expressed as:

```
Rup: estimated throughput from the WLAN interface to the
AP at time k*Tp;

Rdown: estimated throughput from the wired interface to the AP at
time k*Tp;

NL: number of packets lost at the downlink queue in the
time [(k-1)*Tp,k*Tp];

at time k*Tp the TBF rate R is changed according to:

  if NL = 0
  then
    first_loss = true
    R = min (R+Rstep, C_max_theor);
  else
    target_rate = (Rdown + Rup)/2
    if first_loss
    then
       first_loss = false
       R = max( min (R-Rstep , Rup-Rstep), target_rate)
       Tokens = min(Tokens, B_bucket*Th)
    else
       R = max (R-Rstep, target_rate)
```

The parameter R_{step} controls the maximum speed of rate increase and decrease (equal to R_{step}/T_p). Too small values of R_{step} may make difficult the startup of new downstream connection (that need a certain amount of free capacity) and may reduce the efficiency when the bandwidth needs to be increased after a sudden reduction of the capacity required by downlink connection. On the contrary, too large values of R_{step} may give rise to significant throughput oscillations due to interactions with the underlying TCP congestion control mechanisms.

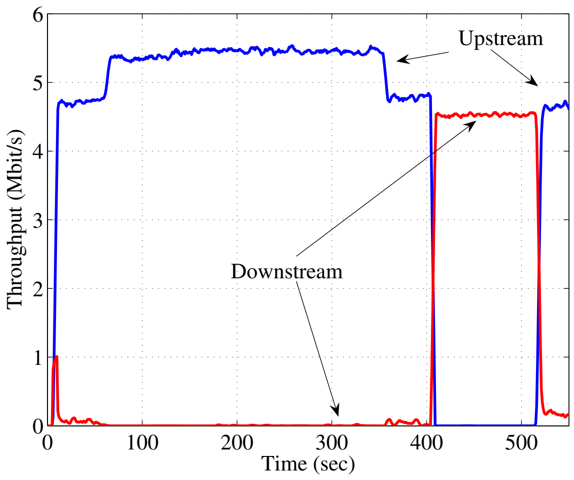

Fig. 8. Time evolution of upstream and downstream throughput without rate control.

In the next sub-section, we evaluate numerically the effectiveness of our solution. We have empirically chosen a value of R_{step} equal to 200 kbit/s, since such choice provides good performance, in our case study. We also point out that this choice is a matter of a trade-off between convergence time and granularity of the mechanism and that our analysis has shown that it is not a critical one, in the sense that the system is loosely sensitive to this parameter.

A. Numerical results

To demonstrate the effectiveness of the adaptive rate limiter, we resort to a time-based analysis, by performing a simulation experiment in which the number of active sources varies with time. The time-schedule of the number of active upstream and downstream connections is reported in Table I. For the same connection activity pattern, we have simulated the system by using three different approaches: i) no rate control; ii) our static rate control with R=2.3 Mbit/s and B_{bucket}=500 packets of 1500 bytes and T_h=0.9; iii) our adaptive rate control algorithm with the parameters reported in Table II.

Fig. 8 reports the temporal evolution of the upstream and downstream throughput, without rate-control. We observe that when there are active upstream connections (i.e., during the intervals 0÷400 and 500÷550 seconds), all downstream connection are starved. In addition, we have analyzed upstream starvation phenomena and registered the occurrence of such phenomena when more than six upstream connections are active (the related numerical results are not reported here for space limitations).

Fig. 9 reports the temporal evolution of the throughput obtained by using the static rate limiter. We note that critical

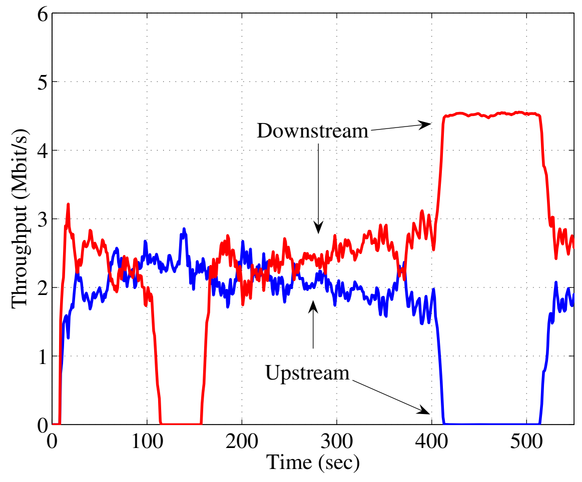

Fig. 9. Time evolution of upstream and downstream throughput with static rate control.

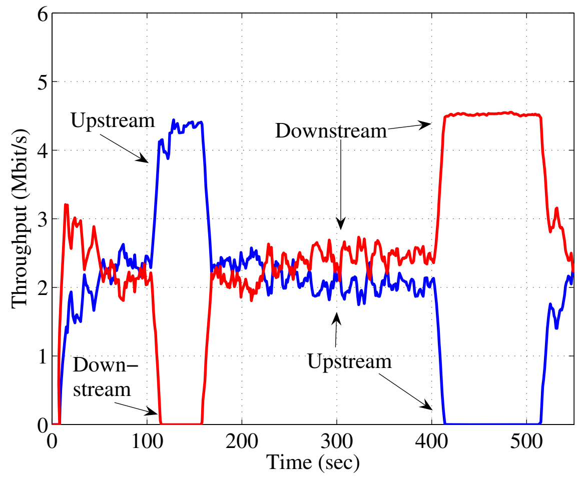

Fig. 10. Time evolution of upstream and downstream throughput with adaptive rate control.

starvation of downstream connection has been avoided. When both upstream and downstream connections are present, their total throughputs are comparable, as expected by the choice of R=2.3 Mbit/s. Nevertheless, this figure shows the necessity of an adaptive mechanism in order to avoid a waste of resources. In fact, during the 100÷150 seconds time interval, when there are no downstream connections, the upstream connections are not able of obtaining more than 2.3 Mbit/s, thus wasting half of the radio capacity.

Finally, Fig. 10 reports the temporal evolution of the throughput obtained by using the adaptive rate limiter. The proposed mechanism is effective in granting all the capacity to the upstream connections during the 100÷150 seconds time interval. Moreover, the sudden throughput decrease and increase, occurring after variations of the number of connections, prove that the reaction of the adaptive control is fast enough and that the resource waste is very limited.

VII. Conclusions and Further Work

In this paper, we addressed fairness issues in a wireless access network based on the IEEE 802.11b standard, operating in DCF mode at 11 Mbit/s. We proposed a solution based on a "rate limiter", operating on the uplink traffic. The rate of the rate limiter can be set statically or dynamically in response to network traffic conditions. Since the rate limiter enforces a limitation on the rate of upstream TCP connections, the remaining WLAN capacity remains available to downstream connections. When the rate is statically set, the system may waste resources, when TCP downstream connections are not greedy, i.e., when they do not use all available capacity.

Our proposed rate limiter mechanism avoids critical starvation in all considered scenarios and is independent of RTTs. Simulation results show the effectiveness of the proposed adaptive control in a dynamic scenario where the number of upstream and downstream connections varies with time. Simulation results are confirmed and validated *in a real ad-hoc developed test-bed* (the test bed and some selected measurements are reported in Appendix IV of [6]).

Coming to possible extensions of this work, a straightforward one is the support of a mix of TCP and UDP traffic (supporting real time services). We are addressing this issue by taking into account the capacity consumed by UDP flows in the adaptive rate limiter setting, either on a pre-reservation basis or on the basis of a dynamic estimation. The solution will also consider a separate queuing of UDP and TCP packets in the access point.

Finally, we note that throughout all this work we assumed an ideal behavior of the IEEE 802.11b radio link. In a more realistic environment, several factors (per-station link rate adaptation, physical layer impairments and transmission errors, MAC layer impairments, such as hidden terminals, etc.) contribute to a time-variant reduction of the WLAN capacity.

References

[1] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC layer fairness in wireless packet networks," in *Proc. ACM Mobicom 2000*.
[2] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless LAN," in *Proc. MobiCom 2000*.
[3] S. Pilosof, R. Ramjee, Y. Shavitt, P. Sinha, "Understanding TCP fairness over wireless LAN," in *Proc. IEEE INFOCOM 2003*.
[4] G. Bianchi, N. Blefari-Melazzi, E. Graziosi, S. Salsano, V. Sangregorio, "Internet access on fast trains: 802.11-based on-board wireless distribution network alternatives," in *Proc. IST Mobile & Wireless Communications Summit 2003*.
[5] S. Shenker and J. Wroclawski, Network Element Service Specification Template, RFC 2216.
[6] N. Blefari-Melazzi, A. Detti, I. Habib, A. Ordine, S. Salsano, "TCP fairness issues in IEEE 802.11 networks: problem analysis and solutions based on rate control," University of Roma Tor Vergata, technical report. Available at ftp://TW05:TW05@160.80.81.106/tw05/TCPfairnesslong.pdf and on request by contacting the authors.
[7] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/
[8] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The effects of asymmetry on TCP performance," *ACM Mobile Networks and Applications (MONET) Journal*, vol. 4, no. 3, 1999.
[9] W. Stevens, TCP Congestion Control, RFC 2001.
[10] S. Floyd *et al.*, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 2582.
[11] M. Mathis *et al.*, TCP Selective Acknowledgement Options, RFC 201.
[12] Linux advanced routing & traffic control - LARTC, http://lartc.org/

Nicola Blefari-Melazzi received his Laurea degree magna cum laude, in Electrical Engineering in 1989, and earned his "Dottore di Ricerca" (Ph.D.) in Information and Communication Engineering in 1994, both at the Universit di Roma, "La Sapienza," Italy. In 1993 he joined the Universit di Roma "Tor Vergata," as an Assistant Professor. From 1998 to 2002 he was an Associate Professor at the Universit di Perugia. In 2002 he returned to Universit di Roma "Tor Vergata," as a Full Professor of Telecommunications, teaching courses in Telecommunications Networks and Foundations of Internet.

Dr. Blefari-Melazzi has been involved in various consulting activities and research projects, including standardization and performance evaluation work. He has participated in and continues to work with a number of European-funded projects including MEDIAN, SECOMS/ABATE and ASSET in the ACTS program, the IST projects SUITED, WHYLESS.COM, ICEBERGS, FIFTH, SATNEX, SIMPLICITY, DISCREET, SATNEX II, E2R II, SMS as well as projects funded by the European Space Agency. In the SIMPLICITY (www.ist-simplicity.org) and in the SMS (www.ist-sms.org) projects, he is the coordinator for the whole consortium. He has been a reviewer for numerous research proposals and has been an auditor for EU and ITEA sponsored projects.

His research interests include the performance evaluation, design and control of broadband integrated networks, wireless LANs and satellite networks. He is also conducting research on multimedia traffic modelling, mobile and personal communications, quality of service in the Internet, ubiquitous computing, reconfigurable systems and networks, service personalization, autonomic computing.

Andrea Detti received the degree in Telecommunication Engineering from the University of Rome "La Sapienza" in 1998. He joined the University of Rome "Tor Vergata" in 1998, where he received in 2001 the Ph.D. degree in microelectronic and telecommunications, since 2004 he is a researcher of the same University. His current research interests are mainly focussed on wireless networking issues.

Ibrahim Habib received a Ph.D. degree in 1991 from the City University of New York, an M.Sc. degree from Polytechnic University of New York, and a B.Sc. degree from Ain Shams University, Cairo, Egypt, all in electrical engineering. From 1981 to 1983 and from 1984 to 1988, he was a computer networks engineer involved in the planning, system engineering, and installation of several IBM SNA networking projects in both Egypt and Saudi Arabia. In 1991 he joined the faculty of the City University of New York where is now a professor. From 1997 to 2001 he was a part-time scientist, first at AT&T and then Telcordia Technologies. His industrial experience spans different areas of data networking including architecture designing of IP/DWDM networks, IP quality of service migration strategies and architecture, and IP traffic engineering. His research interests span different areas of management of resources in IP, wireless, and optical networking. He was a Guest Editor of *IEEE Journal on Selected Areas in Communications* three times in 1997, 2000, and 2003 and a Guest Editor of *IEEE Communications Magazine* in 1995, 1997, and 2006. He also served as an Editor of the same magazine from 1994 to 1997. He was also a Guest Editor of the John Wiley *Journal on Wireless Communications and Mobile Computing* in 2002, and is currently an Editor of the same journal. He has been co-chairman and chairman of many symposia and workshops at various IEEE conferences.

Alessandro Ordine received his "Laurea Specialistica" Degree in Telecommunications Engineering (University of Rome "Tor Vergata") in April 2004 with a thesis on "Controlling TCP Fairness in WLAN access network." Since November 2004, he is a PhD student at University of Rome "Tor Vergata." He has participated in the EU project FIFTH (on internet access via satellite on high-speed trains) and in the Italian PRIN TWELVE (on the service differentiation in 802.11 networks). His current research interests include analysis of TCP and VoIP over Wireless LANs.

Stefano Salsano was born in Rome in 1969. He received his honours degree in Electronic Engineering from the University of Rome (Tor Vergata) in 1994. In 1998 he was awarded a PhD from the University of Rome (La Sapienza). Between the end of 1997 and 2000, he worked with CoRiTeL, a telecommunications research institute, where he was co-ordinator of IP-related research. Since November 2000 he has been an assistant professor ("Ricercatore") at the University of Rome (Tor Vergata) where he teaches the courses on "Telecommunications transport networks" ("Reti di trasporto") and on "Telecommunication networks." He has participated in the EU funded projects INSIGNIA (on the integration of the B-ISDN with Intelligent Networks), ELISA (on the integration of IP and ATM networks, leading the work package on Traffic Control), AQUILA (QoS support for IP networks, leading the work package on Traffic Control), FIFTH (on internet access via satellite on high-speed trains), SIMPLICITY (on simplification of user access to ICT technology, leading the work package on architecture definition), E2R (on reconfigurable radio systems), SMS (on service authoring platforms for mobile services, leading the work package on system architecture). He has also participated in ESA founded projects (ATMSAT, ROBMOD) and in national projects founded by the Italian Ministry of University and Research (MQOS, NEBULA, PRIMO, TWELVE). His current research interests include Ubiquitous Computing and Wireless LANs, IP telephony, Peer-to-peer networking architectures, QoS and Traffic Engineering in IP networks. He is co-author of more than 50 publications on international journals and conferences.