Controlling TCP Fairness in WLAN access networks

Nicola Blefari-Melazzi, Andrea Detti, Alessandro Ordine, Stefano Salsano
Department of Electronic Engineering
University of Rome “Tor Vergata” Via del Politecnico 1, 00133 Rome, Italy
Email: {blefari, andrea.detti, alessandro.ordine, stefano.salsano}@uniroma2.it

***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.**

I. INTRODUCTION

I.A. Motivations and Problem Statement

Deployment of Wireless LANs (WLANs) based on the IEEE 802.11 standard is increasingly on the rise. WLAN products possess a high degree of interoperability, thanks to certifications such as “WiFi” [1]. Wireless LANs are shared media enabling connectivity in the so-called “hot-spots” (airports, hotel lounges, etc.), university campuses, intranet access in enterprises, as well as “in-home” for home internet access among others. In all of the above-mentioned scenarios, WLAN coverage is based on “*Access Points”*, devices that provide the mobile stations with access to the wired network. These scenarios are often called “infra-structured” WLANs to distinguish them from the “ad-hoc” WLANs, where mobile stations talk to each other without using Access Points. In the above scenarios, it is crucial to maintain fairness among the 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 pre-requisite 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 Figure 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.).

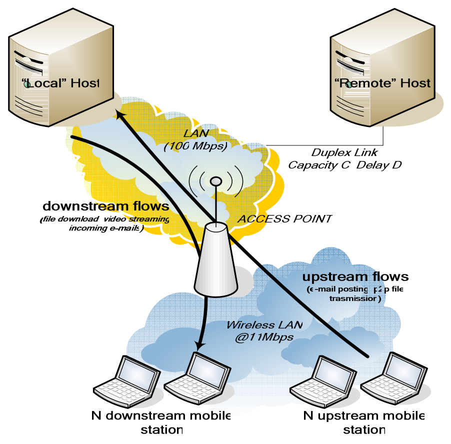

“Local” Host
“Remote” Host
LAN
(100 Mbps)
Duplex Link
Capacity C Delay D
downstream flows
(file download video streaming incoming e-mails)
ACCESS POINT
upstream flows
(e-mail posting p2p file trasmission)
Wireless LAN
@11Mbps
N downstream mobile station
N upstream mobile station

Figure 1: Reference simulation scenario

In the following we introduce the two critical unfairness cases that will be thoroughly analyzed in the next section. In the first case, downstream TCP connections are penalized with respect to upstream ones (see Figure 2). 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 (see Figure 3) 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. This case is also another example of "critical unfairness" which will be explained in more details in the next section.

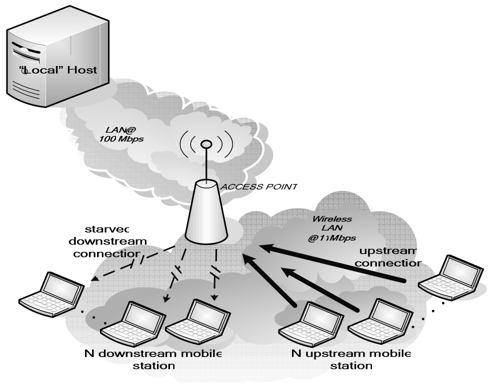

"Local" Host
LAN@ 100 Mbps
ACCESS POINT
Wireless LAN @11Mbps
starved downstream connection
upstream connection
N downstream mobile station
N upstream mobile station

Figure 2: Downstream TCP connection penalized with respect to upstream ones

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.

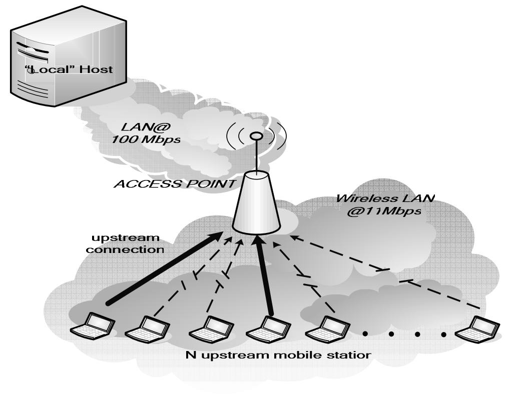


Figure 3: Interaction of several upstream connections

I.B. Related Work and Basic Assumptions

The problem of WLAN fairness at MAC level has been analyzed in several papers, (e.g., [2], [3]); however, as we said earlier, MAC level fairness is not enough. Unfairness among TCP connections within a single WLAN was analyzed in [4]; 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 [4] 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 III). Unfairness among TCP and UDP flows in more complex topologies (i.e., with multiple WLANs) has been preliminary discussed in [5]. 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, implemented via a Token Bucket Filter (TBF) [6]. It 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). The rate-limiter operates on the overall aggregate of uplink packets. The TBF generates tokens at rate R and puts them in the bucket. The rate limiter forwards arriving uplink packets only if there are tokens available in the bucket, otherwise uplink packets are dropped. Each arriving packet consumes a token. The TBF forces packets' loss when the uplink aggregate rate is higher than the TBF rate R and no token is left in the bucket. 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, 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.

As for the organization of the paper, in Section II we discuss the "basic" system model without rate control mechanisms and introduce, evaluate and comment some performance measures. In Section III, we present our solutions based on rate control, to face the so called critical unfairness. The static approach is detailed in Section IV, together with the related performance evaluation; similarly, we present the adaptive rate control in

Section V, together with the related performance evaluation. Finally, in Section VI we discuss conclusions and future work.

II. System Model and Performance Measures

We started our work by analyzing the performance of upstream TCP connections resulting from both mobile stations and fixed hosts, by means of simulations. 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%. The simulation model is shown in Figure 1 above. 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 Figure 1, in our simulation environment the wired host can be connected to the Access Point via a Fast Ethernet (100 Mb/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. We also assumed that TCP sources have always information to transmit (this assumption is commonly denoted in the literature as "greedy source" or "infinite file transfer" model). As for the downlink buffer, we will present simulations in which we vary its size to analyze the impact of this critical parameter (e.g., 50, 100, 300 packets). When it is not otherwise specified, the downlink buffer size is 100 packets, which, according to [4], 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. 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.

II.A. Numerical results

We start our analysis with the case of "no rate-control" without any special mechanisms to improve the performance 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 800 seconds of simulated time; the throughput was measured during the last 200 seconds of each experiment, to avoid transient effects and operate in conditions more near to the steady state.

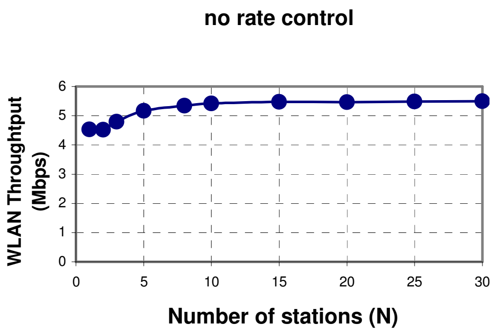


Figure 4: Average total throughput ("local wired host" scenario) without Rate Control Mechanisms

Figure 4 shows the total throughput in the WLAN as a function of N ($N=N_{dn}=N_{up}$), when no rate control is implemented. Figure 5 shows the total upstream and downstream throughputs as a function of N (the total throughput of Figure 4 is the sum of these two components). In Figure 5, 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 (shown in Figure 4) slightly increases with N, as an indirect consequence of the increase in packets' loss in the downlink buffer. When the losses are high, more TCP segments than TCP ACKs are transmitted in the WLAN. In fact, if there is no loss, there will be one TCP ACK transmitted for each TCP segment. 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. This means that the ratio between TCP ACKs and TCP segments decreases as the loss in the downlink buffer increases. 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).

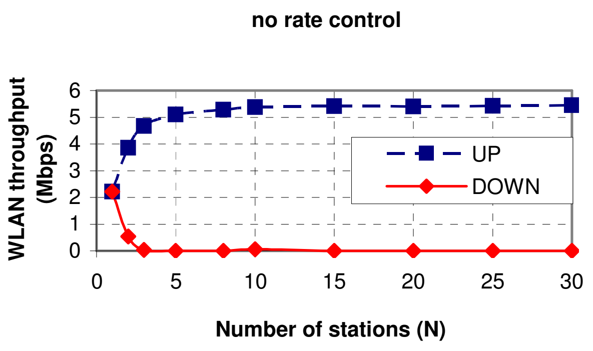


Figure 5: Upstream and downstream throughput ("local wired host" scenario) without rate control mechanisms

We focus now on upstream connections. Figure 6 reports the throughput perceived by each upstream connection for *N*=5, 10 and 20 (always in the "no rate-control" case). It is evident that there is no fairness 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. The bar charts show that for *N*=5 all flows are able to start. For *N*=10 and *N*=20 only 6 and 8 flows, respectively, actually use the WLAN resources. As anticipated, the ratio σ_u/R_u is a good gauge of unfairness and, as shown in Figure 7, it sharply increases for N>5.

II.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.

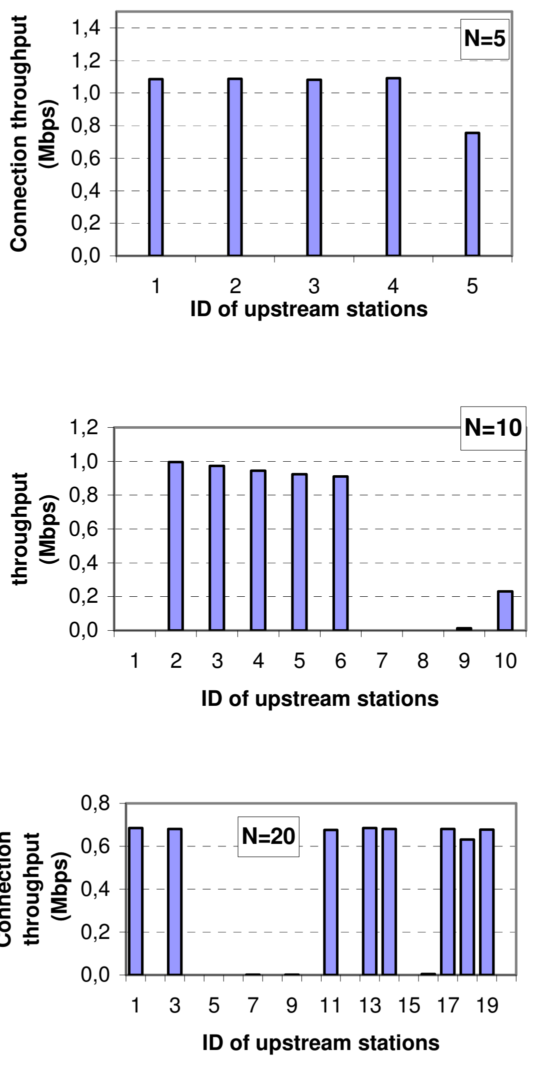


Figure 6: Throughput of upstream connections ("local wired host scenario")

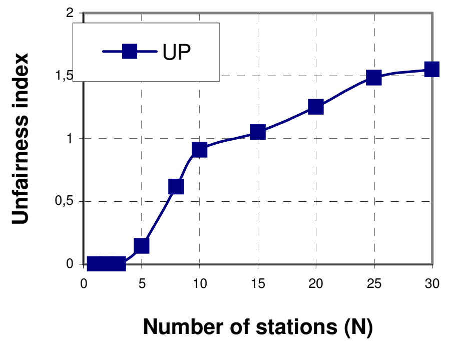


Figure 7: Ratio σ_{up}/R_{up} for upstream connections ("local wired host scenario")

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.

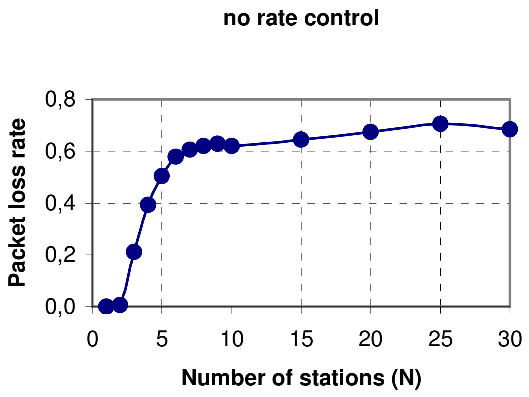


Figure 8: Packet loss rate in the downlink access point buffer ("local wired host" scenario)

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, as shown in Figure 5. To support this hypothesis, let us look at Figure 8 which shows the packet loss probability in the downlink buffer of the Access Point as a function of N. 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). 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., Figure 8 shows that the loss probability is in the order of 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 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 behavior illustrated in Figure 6. 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, 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. Figure 9 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·N·CW_max, where CW_max is the TCP maximum congestion window size (expressed in packets of 1500 bytes, following NS-2 conventions). Obviously, the loss probability decreases when the buffer size increases. Consequently, the number of stations representing the critical threshold beyond which such starvation occurs increases too. 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 (see Figure 9). 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).

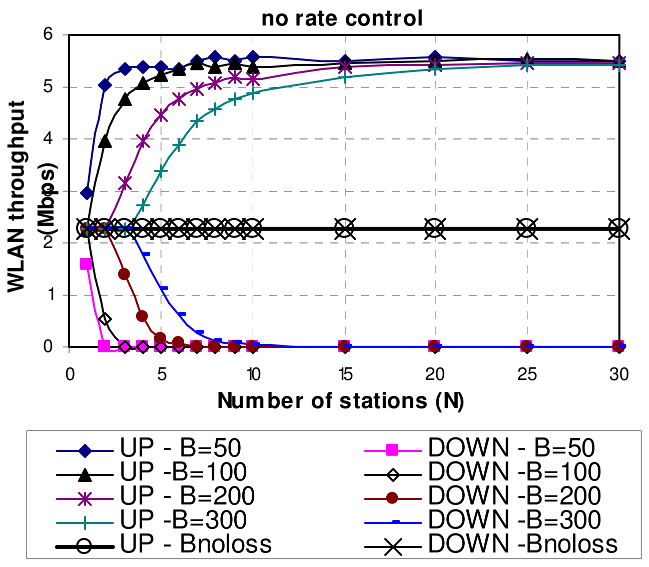


Figure 9: Total upstream and total downstream throughput

Therefore, once all TCP segments are in transit, the communication goes on into a "Stop&Wait" fashion (i.e., a generic TCP source can transmit only one segment and then must wait for the corresponding TCP ACK). The downlink TCP packets and the TCP ACKs for the uplink flows get stored in the downlink buffer in the Access Point. When the Access Point sends a downlink TCP packet, the corresponding "downstream" mobile station is enabled to send the TCP ACK. When the Access Point sends a TCP ACK for an uplink flow, the corresponding "upstream" mobile station is enabled to send a new TCP segment. Hence, the Access Point gets half of the overall bandwidth for the downlink transmission while the stations equally share the remaining bandwidth for their uplink transmission. Hence, it is easy to conclude that under these conditions all TCP connections get the same amount of bandwidth in the steady state. This is shown in Figure 10 and Figure 11, where we plot the throughput of individual connections. To verify our conjecture about the "Stop&Wait" behavior, we have complemented our throughput measurements (shown in Figure 9) by analyzing what happens at the MAC level under the same conditions. For instance, for N=15, with a downlink buffer size equal to B_{noloss}, we have found that the mean number of active stations at MAC level (i.e., stations that are transmitting or with backlogged traffic at the MAC layer), excluding the access point is only 0.76. Considering that the access point is always active, this means that communication is basically taking place one at a time (i.e., between the access point and one mobile station at a time). This implies that the WLAN is operating as if a polling scheme is in place and capacity is shared evenly among all stations. On the other hand, when the buffer size is smaller than B_{noloss}, we have found that more than one station have packets stored in the MAC queue.

These stations compete for medium capacity. There is no polling effect that can be observed and unfair access occurs. As a matter of fact, with a buffer size of 100 packets, we have found that the mean number of active stations at the MAC level is 3.16 in addition to the Access Point, thus proving our conjecture (note that with N=15, there will be 7 stations that are able to send their upstream connections, while all the other ones are starved).

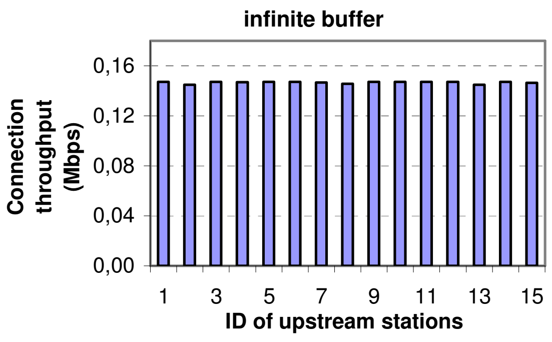


Figure 10: Fairness achieved in lossless conditions for upstream stations

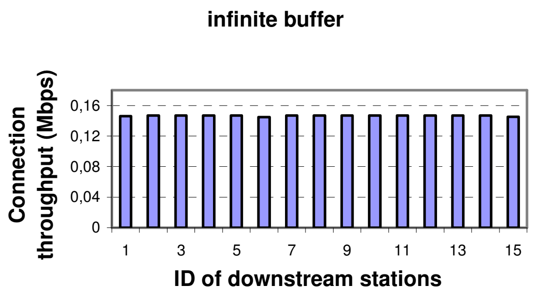


Figure 11: Fairness achieved in lossless conditions for downstream stations

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}. This choice would also have the advantage of guaranteeing to all connections the same throughput, thus reaching a perfect fairness [4]. However, increasing the size of the downlink buffer has the disadvantage of increasing the queuing delay, with obvious consequences on the overall Round Trip Time experienced by TCP connections. 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 Mb/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. This effect is not apparent in our figures since we are focusing on long-lived TCP ones.

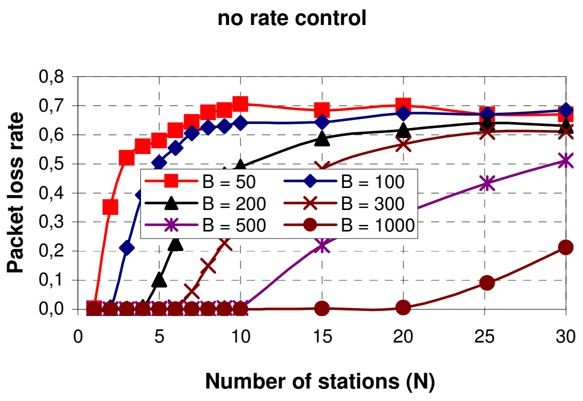


Figure 12: Packet loss rate in the downlink Access Point buffer ("local wired host" scenario)

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. The same settings of Figure 9 were used for Figure 12 to evaluate the downlink buffer loss rate as a function of N, and for different values of the buffer size. 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. In the previous sub-section, we stated that the total throughput increases with N, and we anticipated that this would happen because as N increases the ratio between TCP ACK segments and overall TCP traffic decreases (see Figure 4). This phenomenon can be further explained with the ensuing considerations. For small values of the downlink buffer size, as N increases, the downstream connections tend to starve. The radio link capacity is used only by some upstream connections, and the AP downlink buffer contains mainly ACK segments.

Additionally, as the downlink buffer loss probability increases, the ratio between the number of overall TCP segments and the number of ACKs exchanged over the air interface increases as well (as explained in our comments to Figure 4 in Section II.A). The segments are significantly longer (1500 bytes) than the related ACKs (40 bytes), thus, the overhead at the MAC level decreases and the total throughput increases (from 4.5 Mbps to 5.3 Mbps for the range of N shown in Figure 4). It is worth noting that the value of 4.5 Mbps is the maximum total throughput obtainable in any *lossless* scenario. In our scenario, TCP is more efficient for large values of the ACK loss probability. In fact, under these conditions a throughput of 5.3 Mbps is reached (see Figure 4). On the other hand the advantage of enjoying an increased throughput has to be traded off with the above mentioned cons and in particular with the risk of heavy unfairness and connections starvation.

Before concluding the Section, we make two additional considerations:

1) Starvation phenomena are related to the TCP startup phase; this means that also other TCP versions will experience similar problems.

2) All of the above discussed unfairness phenomena happen when the overall system bottleneck is the WLAN radio interface. If the bottleneck is somewhere else in the fixed network, TCP connections will not be able to grab all the WLAN capacity and the WLAN will not introduce unfairness.

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

III. 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 [4] 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 happening in the Access Point). In this case, fairness conditions are achieved since, as discussed in Section II.B, each source operates in a "stop & wait" mode. 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 at the IP level, before the uplink buffer. 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 Mb/s), and 2) the bucket size B_{bucket} (expressed in Mbit). 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. 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 IV and V, respectively.

IV. STATIC RATE CONTROL

The Figure 13 shows the static rate-limiter as it would be implemented within the access point. However, the rate-limiter could also be implemented in an external device (e.g., a router) connected to the access point. This alternative 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. Such solution is depicted in Figure 14 .

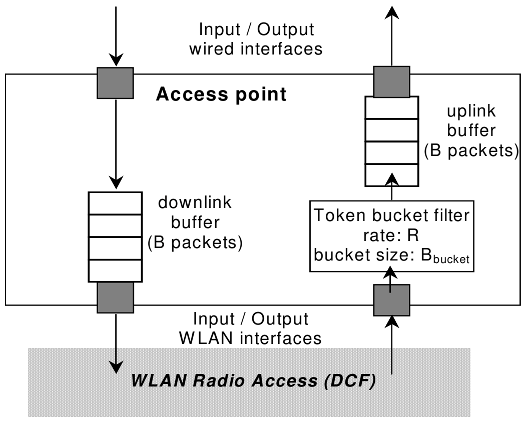


Figure 13: Rate control solution based on a rate-limiter

In the following we will first present a subset of results of a simulation study having the aim of selecting the parameters of the Token Bucket filter, i.e., the rate R and the bucket size B_{bucket}. Ideally, we want to choose such parameters so as to avoid starvation and to provide a fair access possibility to all applications, while maximizing the overall throughput. Once this is done, we will analyze the performance of our proposed solution and compare it with the approach proposed in [4], which can be considered as the current state of the art. We started by studying the effect of the parameter R on the performance and we found out that there is a range of R where the throughput is maximized and fairness is achieved. In particular, we measured the total upstream and downstream throughput by varying the rate R from 1.3 Mb/s to 3.3 Mb/s, for several values of the number of sources $N_{dn}=N_{up}=N$. Such throughputs are plotted in Figure 15 as a function of R for N=3 and N=15, as an example (other values of N give rise to the same behavior). These results show that capacity is never wasted by varying the rate R, as the total throughput is constant. The token bucket rate limiter enforces an upper bound to the upstream throughput. The upstream throughput is closer to such bound when the number of sources is large. For instance, for R=2.3 Mb/s the upstream throughput is 2.25 Mb/s for N=15 and 2 Mb/s for N=3. This behavior can be explained by considering that the TCP connections are operating in congestion avoidance, and their throughput fluctuates in time. When the rate limiter introduces packet losses, the TCP upstream connections suddenly reduce their throughput. Afterwards, they slowly increase it again. These throughput values fluctuate around (but mostly below) a value equal to R/N. For small values of N, the size of oscillations is higher than for high values of N and the gap between the upstream throughput and the token bucket rate is correspondingly larger, since the token bucket rate limiter drops comparatively more packets. Thanks to this analysis, we can choose a value for the rate R, with the obvious aim of providing good fairness performance. As a case study, in this paper we set R=2.3 Mb/s. We performed a similar analysis to select the bucket size B_{bucket}.

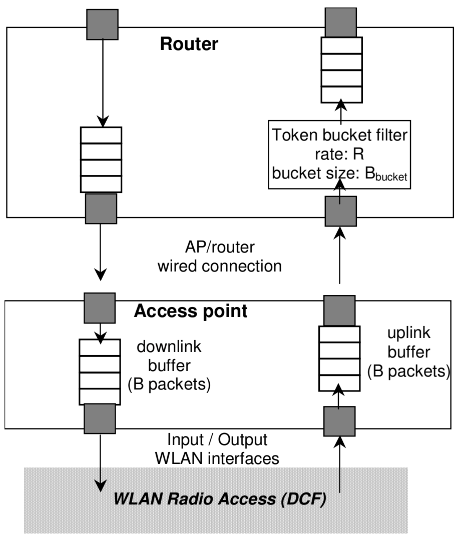
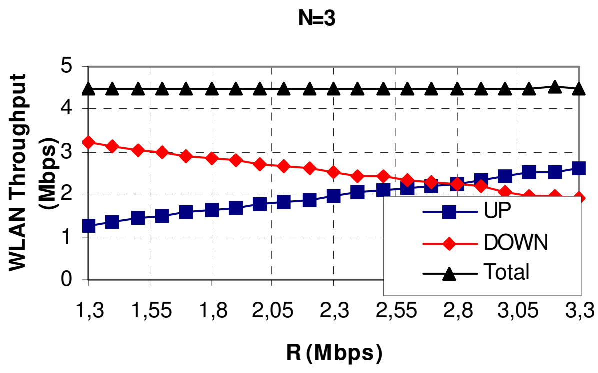


Figure 14: Rate-limiter placed on an external device

We evaluated by means of simulations the upstream throughput as a function of B_{bucket} for several values of $N_{dn}=N_{up}=N$ and for several values of the rate R.



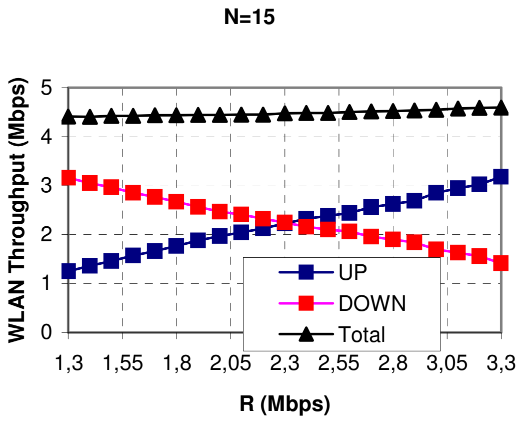


Figure 15: Total, upstream and downstream throughput versus rate limiter R (for N=3 and N1=15)

In Figure 16, we report a small selection of these simulation results. The upstream throughput is plotted for N=3 and N=15, and for two values of R: 2 Mb/s (the two lowermost curves) and 3 Mb/s (the two uppermost curves). We observe that the upstream throughput gets closer to the bound of the token bucket rate as the bucket size increases. However, the derivative of such curve decreases with the bucket size. As a case study, in this paper we set B_{bucket}= 200 packets (of 1500 bytes). Summing up, in the following we will set the parameters of the Token Bucket filter to R=2.3 Mb/s and B_{bucket}=2.4 Mb (corresponding to 200 packets of 1500 bytes).

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 II, 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 [4], iii) our proposed static rate limiter.

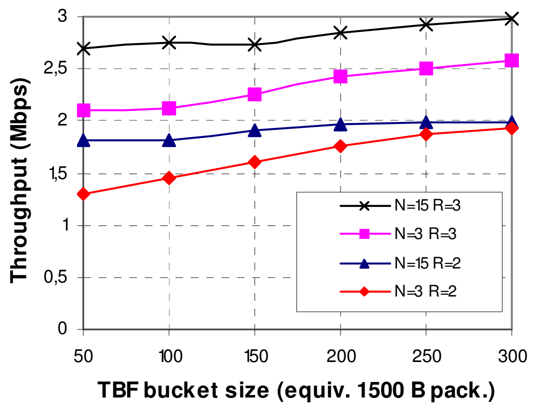


Figure 16: Upstream throughput as a function of the bucket size

This comparison is reported in Figure 17, which shows the total throughput as a function of *N*. The lossless rate control of [4] 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 Figure 18. The lossless rate control of [4] 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. We will come back to this effect later on in this Section. Here we just observe that we have obtained the important result of avoiding the so-called critical unfairness, observed in the "no rate control" case, without the disadvantages of the lossless rate control in terms of complexity. To continue our analysis, 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 (the so-called unfairness index). We recall that if the ratio σ_{up}/R_{up} is equal to zero, then we have perfect fairness; otherwise the greater the ratio, the greater the unfairness. This ratio is shown in Figure 19 as a function of *N*; the lowermost curve reports this ratio evaluated for the case of our proposed solution, the other curve is relevant to the same scenario but without rate control.

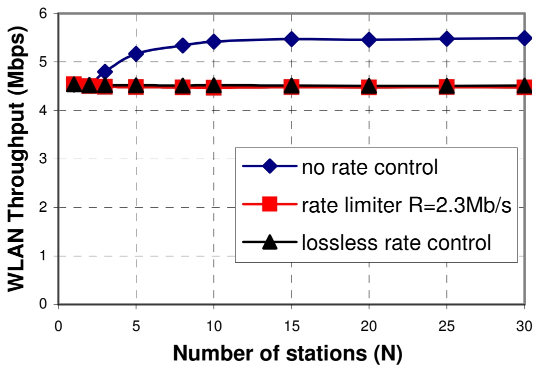


Figure 17: Average total IP level throughput

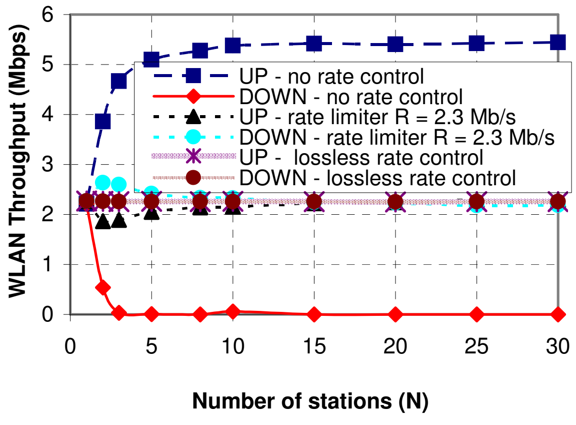


Figure 18: Upstream and downstream throughput

The ratio relevant to the lossless rate control proposed in [4] is not plotted since it is always equal to zero for any *N* (perfect fairness). Our solution shows small unfairness among upstream flows (in this case the unfairness increases with *N*), which can be considered as tolerable, given the relatively small price paid to obtain it (in terms of added system complexity); in any case, our solution performs better than the "no rate-control" case.

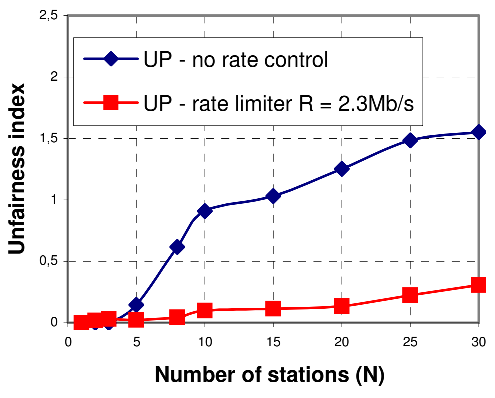


Figure 19: Upstream fairness

We observe that our static rate-limiter is always capable of avoiding the critical unfairness, which is instead evident in the case of no-rate control. 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. However, the "per connection" fairness was not among our goals. We are satisfied as long as the critical unfairness is avoided and resources are not wasted by using a low complexity solution. 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 [4], is not affected by limitations related to the RTT. In this simulation study (reported in APPENDIX I) we consider the "remote wired host" scenario (see the right part of Figure 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 [4] 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 we have attained our goals in terms of fairness. However, the static setting of the rate R implies a possible waste of WLAN resources when downstream applications are not greedy or not active. In fact, in this case the Token Bucket Filter unnecessarily limits the upstream connections. Therefore, we improved our solution by introducing a mechanism to control the setting of the rate-limiter, by adapting it to the downstream traffic. Such adaptive rate control is presented in Section V, together with the related performance evaluation.

V. Adaptive Rate Control

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 $R_{\mathrm{down}} < C\text{-}R$, then the static rate limiter causes a waste of capacity in the order of $C\text{-}R\text{-}R_{\mathrm{down}}$. The idea of the adaptive rate control is to increase the rate of the token bucket filter in these conditions up to $R'=C\text{-}R_{\mathrm{down}}$ 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} (Mb/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 in general considers information such as the state of the downlink queue, and the estimated uplink and downstream throughputs. The architecture of the adaptive rate limiter is depicted in Figure 20. There are several degrees of freedom in the choice of the control rule. The optimization of such choice is out of the scope of this work. Here, we have chosen a case-study control rule, and tested it under some realistic scenarios in order to show the viability of our approach. In particular, we chose a control rule based upon the estimation of uplink and downlink traffic and not upon the state of the access link downlink buffer. Therefore, it is possible to implement the adaptive rate limiter solution also in an external device (see Figure 21).

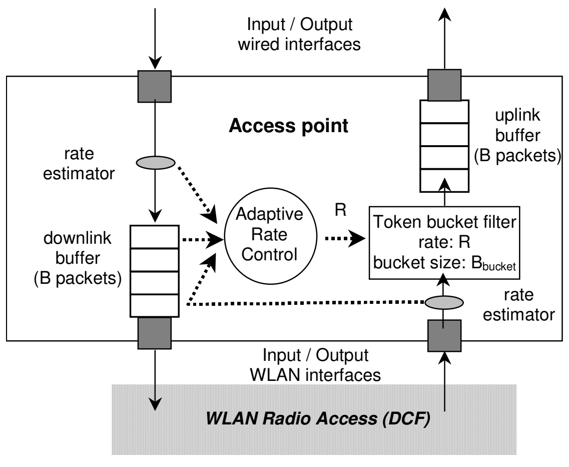


Figure 20: Architecture of the adaptive rate-limiter

Our proposed adaptive mechanism works as follow: we estimate the "instantaneous" throughput (here we use the same definition of throughput given in Section II, 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 II), which is very simple to implement. If the total estimated throughput (i.e., $R_{\mathrm{up}} + R_{\mathrm{down}}$) is greater than $R_{\mathrm{max}} = 4.5$ Mbps, then we assume that packet losses are occurring in the downlink buffer. Consequently, the token bucket rate must be reduced to avoid starvation of TCP connections, due to high loss.

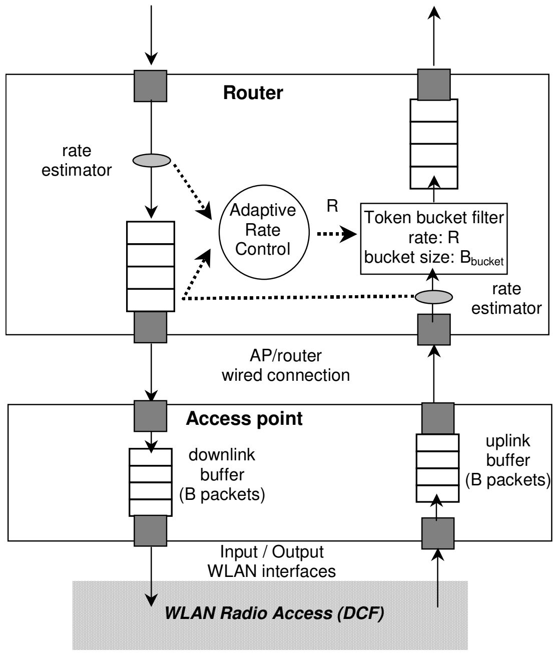


Figure 21: Adaptive rate-limiter using an external device

On the contrary, if $R_{\mathrm{up}} + R_{\mathrm{down}} < R_{\mathrm{max}}$, then the token bucket rate can be increased to avoid wasting of resources. The adaptive algorithm, which runs periodically at the time instants $T=kT_{\mathrm{p}}$, can be expressed as:

```
let $R_{up}$ the estimated throughput coming to the AP from the WLAN interface at time $k \cdot T_p$;
let $R_{down}$ the estimated throughput coming to the AP from the wired interface at time $k \cdot T_p$;
at time $k \cdot T_p$ the TBF rate R is changed according to:
        if (Rup+Rdown) > Rmax
                then R = max (Rmin, R-Rstep);
                else R = min (Rmax, R+Rstep);
```

The parameters R_{min} and R_{max} have to be chosen so as to avoid starvation and wasting of resources. R_{max} is set to 4.5 Mbps which, as discussed in Section IV is the maximum rate that can be achieved in a lossless situation. R_{min} is set to 2.3 Mb/s, as done in section IV. Note that the R_{min} value may be seen as the minimum uplink guaranteed rate. The difference between the overall capacity and $R_{\min}$ may be seen as the minimum downlink guaranteed bandwidth. This means that uplink/downlink bandwidth asymmetries can be opportunely tuned by suitably regulating the parameter $R_{\min}$. 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.

In the next sub-section, we evaluate numerically the effectiveness of our solution. We have empirically chosen a value of R_{step} equal to 200kb/s, since such choice provides good performance, in our case study.

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 1. 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}=200 packets of 1500 bytes; iii) our adaptive rate control algorithm with the parameters reported in Table 2.

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 downstream	3	3	0	3	6	6	10	10	10	10	10
No. active upstream	3	10	10	10	10	6	6	6	0	0	3

Table 1: Time-schedule of the simulation experiments

Parameter	Value
AP downlink buffer B (packets)	100
TBF bucket size B_{bucket} (bytes)	200*1500
$R_{\min}$ (Mbps)	2.3
$R_{\max}$ (Mbps)	4.5
R_{step} (kbps)	200
T_p (ms)	500 ms

Table 2: Adaptive rate limiter parameters

Figure 22 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).

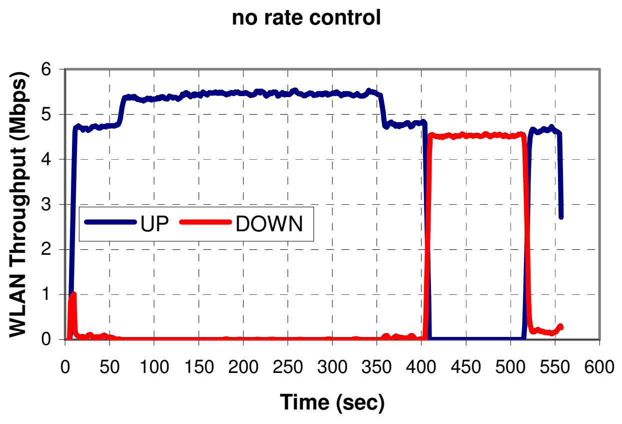


Figure 22: Time evolution of upstream and downstream throughput without rate control

Figure 23 reports the temporal evolution of the throughput obtained by using the static rate limiter. Notice that critical 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 Mbps.

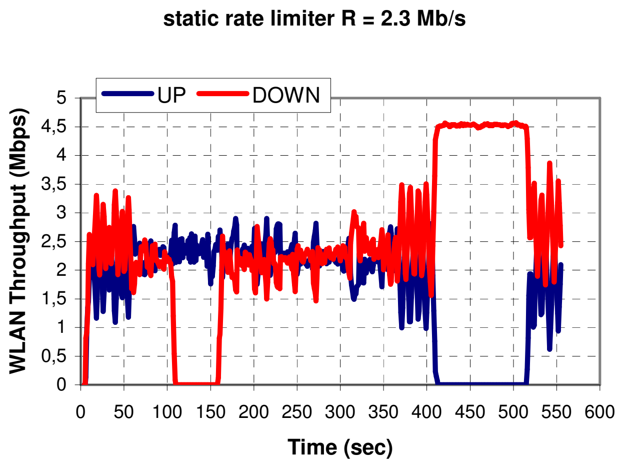


Figure 23: Time evolution of upstream and downstream throughput with static rate control.

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 Mbps, thus wasting half of the radio capacity. Finally, Figure 24 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 (i.e., at time instants 100, 150, 450, 500 sec), prove that the reaction of the adaptive control is fast enough and that the resource waste is very limited.

VI. 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 Mbps. We have shown that without suitable countermeasures, connections use the available capacity in a very unfair way. We also identified "critical unfairness" that corresponds to complete starvation of TCP connections. We proposed a solution based on a "rate limiter", operating on the uplink traffic. The rate limiter is implemented by means of a token bucket filter and it indirectly controls the aggregate rate of upstream TCP connections by relying upon the TCP congestion control mechanisms. The most important setting for the proposed mechanism is the rate *R* of the token bucket filter. Such rate 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.

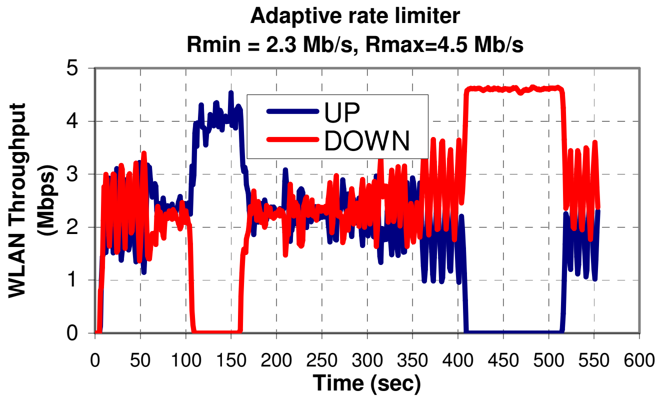


Figure 24: Time evolution of upstream and downstream throughput with adaptive rate control

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 an ad-hoc developed test-bed.

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.

Acknowledgment

This work has been partially founded by Italian research project TWELVE

VII. APPENDIX I

If the server is connected to the WLAN via an high speed Local Area Network, the round trip time is not an issue. In this case the lossless rate control is an optimal solution, and it represents the upper bound of the performances. Here, we want to analyze what happens when connections experience an higher Round Trip Time. This is mentioned as an open issue in [4]. As we will show, in this case our proposed rate limiter solution may work better then the lossless rate control. We consider now the "remote scenario" (see the right part of Figure 1) and terminate the TCP connections on a remote host in the fixed network, considering different values of the RTT of the link between the Access Point and the wired host. Figure 25 and Figure 26 show the total, upstream and downstream throughput as a function of the number of stations, for an RTT value of 200 and 400 ms, comparing the two solutions: "lossless" rate control proposed in [4], and our static rate limiter.

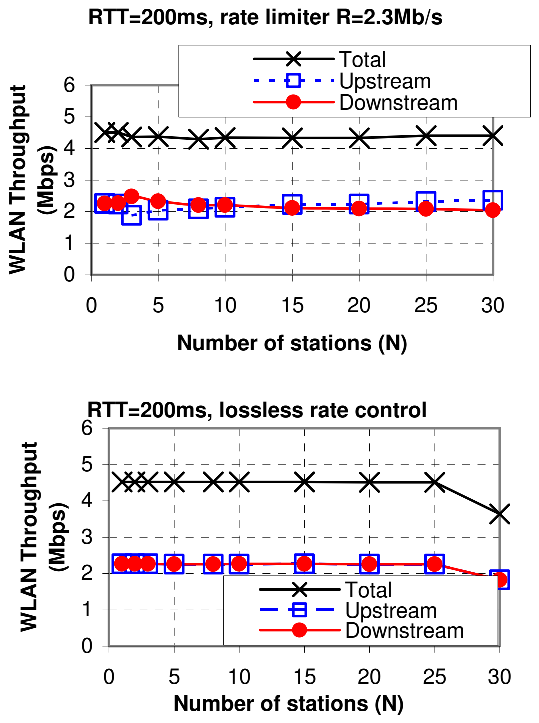
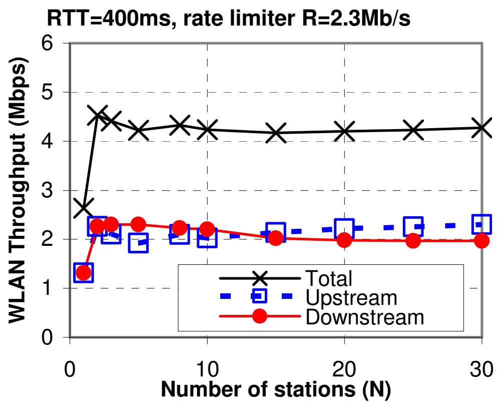


Figure 25: Throughput versus N in the "remote wired host" scenario (RTT=200); comparison between lossless rate control and static rate limiter

For an RTT of 200 ms the results are still comparable with the scenario where the wired host is locally connected to the Access Point. The only difference appears when the number of stations is high (N=30): the lossless rate control is not able to reach the maximum throughput. For an RTT of 400 ms, the performance of the lossless rate control is definitely worse, as it never reach the maximum WLAN throughput. This is due to the imposed limitation on the TCP Congestion Windows of the TCP connections.



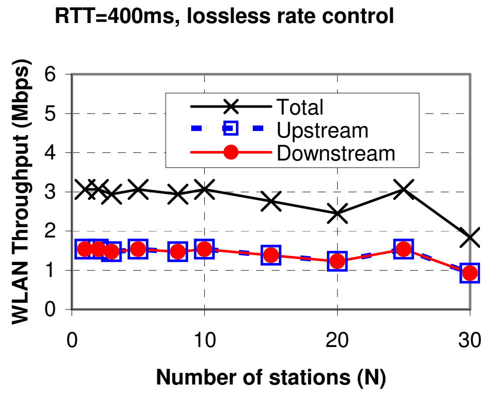


Figure 26: Throughput versus N in the "remote wired host" scenario (RTT=400); comparison between lossless rate control and rate limiter

The well know throughput limit of the window based protocols (Throughput≤*W*/RTT), where *W* is the window size, comes into play and reduces the achievable total throughput. On the other hand, the rate limiter solution is not affected by the increase in the Round Trip Time, both in terms of the total throughput and in terms of upstream/downstream fairness. Figure 27 provides another insight on the RTT problem, as it reports the total throughput versus RTT for the "no rate control", lossless rate control, and the rate limiter solution for a fixed number of connections (N=15). The throughput in case of lossless rate control starts to decrease from RTT≅250 ms.

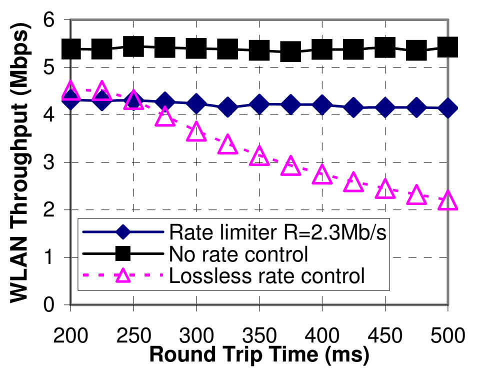


Figure 27: Total throughput vs. RTT, N=15 ("remote wired host" scenario)

VIII. APPENDIX II

When a packet arrives at a rate estimator, an EWMA (exponentially weighted moving average) algorithm is used to estimate the rate R as follows:

```
T : arrival time of the new packet
L : size of the packet (expressed in bits)
$R_{old}$ : previous value of the EWMA estimate
$T_{old}$ : previous time of update of the EWMA estimate
    $R = L/\min(T-T_{old}, \tau)\cdot\alpha + R_{old}\cdot(1-\alpha)$
    $T_{old} = T$
    where $\tau$=600 ms, $\alpha = \min((T-T_{old})/\tau, 1)$
```

where τ represent a time constant which can be set in the order of magnitude of the ratio buffer size/channel capacity, for example considering 100 packets of 1500 bytes at 2 Mb/s leads to 600 ms.When the rate estimate is used at an instant T, it can be updated considering the elapsed time since its last update:

```
T : current time when the rate estimation is used
    $R_{updated} = R\cdot(1-\alpha)$
    where $\alpha = \min((T-T_{old})/\tau, 1)$
```

REFERENCES

[1] Wi-Fi System Interoperability Test Plan, Wireless Ethernet Compatibility Alliance, Version 1.1a, December 11 2001

[2] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC layer Fairness in Wireless Packet Networks", ACM Mobicom 2000, Boston, USA, Aug. 2000

[3] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN", MobiCom 2000, Boston, USA, Aug. 2000

[4] S. Pilosof, R. Ramjee, Y. Shavitt, P. Sinha, "Understanding TCP fairness over Wireless LAN", IEEE INFOCOM 2003, March 30-April 3, 2003, San Francisco, USA.

[5] 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", IST Mobile & Wireless Communications Summit 2003, 15-18 June 2002, Aveiro, Portugal

[6] S. Shenker, J. Wroclawski, "Network Element Service Specification Template", RFC 2216, September 1997

[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/