UNIVERSITÀ DEGLI STUDI DI ROMA

"TOR VERGATA"

FACOLTA' DI INGEGNERIA

DOTTORATO DI RICERCA IN INGEGNERIA DELLE TELECOMUNIVAZIONI E MICROELETTRONICA

CICLO XX

Multi-layer Traffic Control for Wireless Networks

Alessandro Ordine

Docente Guida/Tutor: Ing. Stefano Salsano

Coordinatore: Prof. Nicola Blefari-Melazzi

Summary

Wireless LANs, as they have been defined by the IEEE 802.11 standard, are shared media enabling connectivity in the so-called “hot-spots” (airports, hotel lounges, etc.), university campuses, enterprise intranets, as well as “in-home” for home internet access.

With reference to the above scenarios, WLANs are commonly denoted as “infra-structured” in the sense that WLAN coverage is based on “Access Points” which provide the mobile stations with access to the wired network. In addition to this approach, there exists also an “ad-hoc” mode to organize WLANs where mobile stations talk to each other without the need of Access Points.

Wireless LANs are typically connected to the wired backbones (Internet or corporate intranets) using a wired infrastructure. Wireless Infrastructure Mesh Networks (WIMN) may represent a viable and cost-effective alternative to this traditional wired approach. This is witnessed by the emergence and growth of many companies specialized in the provisioning of wireless infrastructure solutions, as well as the launch of standardization activities (such as 802.11s).

The easiness of deploying and using a wireless network, and the low deployment costs have been critical factors in the extraordinary success of such technology. As a logical consequence, the wireless technology has allowed end users being connected everywhere – every time and it has changed several things in people’s lifestyle, such as the way people work, or how they live their leisure time (videoconferencing, instant photo or music sharing, network gaming, etc.).

On the other side, the effort to develop networks capable of supporting ubiquitous data services with very high data rates in strategic locations is linked with many technical challenges including seamless vertical handovers across WLAN and 3G radio technologies, security, 3G-based authentication, unified accounting and billing, consistent QoS and service provisioning, etc.

My PhD research activity have been focused on multi-layer traffic control for Wireless LANs. In particular, specific new traffic control solutions have been designed at different layers of the protocol stack (from the link layer to the application layer) in order to guarantee i) advanced features (secure authentication, service differentiation, seamless handover) and ii) satisfactory level of perceived QoS. Most of the proposed solutions have been also implemented in real testbeds.

This dissertation presents the results of my research activity and is organized as follows: each Chapter presents, at a specific layer of the protocol stack, a traffic control mechanism in order to address the introduced above issues.

Chapter 1 and Charter 2 refer to the Transport Layer, and they investigate the problem of maintaining fairness for TCP connections. TCP unfairness may result in significant degradation of performance leading to users perceiving unsatisfactory Quality of Service. These Chapters describe the research activity in which I spent the most significant effort. Chapter 1 proposes a simulative study of the TCP fairness issues and two different solutions based on Rate Control mechanism. Chapter 2 illustrates an analytical model of the TCP fairness and derives a framework allowing wireless network providers to customize fairness policies.

Chapter 3 focuses on the Application Layer and it presents new traffic control solutions able to guarantee secure authentication in wireless inter-provider roaming scenarios. These solutions are an integral part of the UniWireless framework, a nationwide distributed Open Access testbed that has been jointly realized by different research units within the TWELVE [38] national project.

Chapter 4 describes again an Application Layer solution, based on Session Initiation Protocol to manage user mobility and provide seamless mobile multimedia services in a heterogeneous scenario where different radio access technologies are used (802.11/WiFi, Bluetooth, 2.5G/3G networks).

Finally Chapter 5 refers to the Data Link Layer and presents a preliminary study of a general approach for routing and load balancing in Wireless Infrastructure Mesh Network. The key idea is to dynamically select routes among a set of slowly changing alternative network paths, where paths are created through the reuse of classical 802.1Q multiple spanning tree mechanisms.

Acknowledgements

My research activity has been carried out at DIE, Università di Roma “Tor Vergata”. I would like to thank all the networking group members in Roma, especially Prof. Stefano Salsano for his patience and for everything he taught me.

The results in the thesis have been obtained in collaboration with other people:

Prof. Giuseppe Bianchi, Prof. Nicola Blefari-Melazzi, Prof. Mauro Brunato, Prof. Andrea Detti, Fabio Feuli, Julian Gutiérrez, Andrea Polidoro, Prof. Stefano Salsano, Danilo Severina and Prof. Luca Veltri. My special thanks to all of them.

Finally I am grateful to my parents and Laura.

Contents

Summary II
Acknowledgements II
Contents III

1 TRANSPORT LAYER: TCP FAIRNESS ISSUES - PROBLEM ANALYSIS AND SOLUTIONS BASED ON RATE CONTROL 1

1.1 INTRODUCTION 1
1.1.1 Motivations and Problem Statement 1
1.1.2 Related Work and Basic Assumptions 3
1.2 SYSTEM MODEL AND PERFORMANCE MEASURES 5
1.2.1 Numerical results 7
1.2.2 Understanding the results 9
1.3 LIMITER BASED RATE CONTROL 15
1.4 STATIC RATE CONTROL 17
1.4.1 Numerical results 19
1.4.1.1 Validation of the Static Rate Limiter for Different RTT 20
1.5 ADAPTIVE RATE CONTROL 22
1.5.1 Numerical Results 25
1.6 ANALYSIS WITH SHORT-LIVED SOURCES 27
1.6.1 Short-lived TCP sources 28
1.6.2 Mix of short-lived and greedy sources 29
1.7 PERFORMANCE EVALUATION IN A REAL TEST-BED 32
1.7.1 Test-bed results without rate-limiter 34
1.7.2 Test-bed results with rate-limiter 36
1.8 CONCLUSIONS 39

2 TRANSPORT LAYER: ANALYTICAL MODEL OF TCP FAIRNESS AND CUSTOMIZABLE FRAMEWORK FOR TRAFFIC CONTROL 40

2.1 INTRODUCTION 40
2.2 CONTRIBUTION OF OUR WORK 42
2.2.1 Fairness model 42
2.2.2 Technique adopted to enforce the fairness 43
2.3 802.11 TCP FAIRNESS MODEL 43
2.3.1 Derivation of p, Q_{ap} and Q_i 46
2.3.1.1 Lossless AP buffer 47
2.3.1.2 Lossy AP buffer 51
2.3.1.3 Qualitative insights on the average occupancy of the STA buffer 54
2.4 CUSTOM FRAMEWORK FOR TRAFFIC CONTROL (CFTC) 54
2.5 ACHIEVING STA FAIRNESS THROUGH CFTC 57
2.6 TESTBED 58
2.6.1 Testbed setup 58
2.6.2 Testbed results 61
2.6.2.1 Impact of the number of downstream TCP connections per STA 62
2.6.2.2 Impact of the number of upstream TCP connections per STA 64
2.6.2.3 Impact of the downstream TCP connection receiver windows 66
2.6.2.4 Impact of the upstream TCP connection receiver windows 68
2.6.2.5 Coexistence of upstream and downstream TCP connections 70
2.6.2.6 UDP performance in presence of TCP connections 71
2.7 CONCLUSIONS 73

3 APPLICATION LAYER: SIP BASED SOLUTION FOR WIRELESS INTER-PROVIDER ROAMING 74

3.1 INTRODUCTION 74
3.2 BACKGROUND AND STATE OF THE ART 75
3.2.1 Authentication systems and techniques 76
3.2.2 Distributed access framework 77
3.3 UNIWIRELESS: SYSTEM PHILOSOPHY AND ARCHITECTURE 78
3.3.1 Uni-Fy gate 78
3.3.1.1.1 Gateway 79

3.3.1.1.2 Gatekeeper 80
3.3.1.2 Authentication Server 80
3.3.1.3 Authentication procedure: Captive portal 81
3.3.2 A novel SIP based authentication procedure 82
3.3.2.1 SIP Authentication overview 82
3.3.2.1.1 Digest Authentication 83
3.3.2.1.2 AKA Authentication 84
3.3.2.1.3 DIGEST – AKA Authentication 86
3.3.2.2 SIP-based authentication method 88
3.3.2.3 Security considerations 92
3.4 UNIWIRELESS IMPLEMENTATION 92
4 APPLICATION LAYER: SIP BASED SOLUTION FOR MOBILITY MANAGEMENT 95
4.1 INTRODUCTION 95
4.2 PROPOSED SOLUTION 97
4.2.1 From SIP Session Border Controllers to mobility management server 98
4.2.2 The Mobility Management Client in the Mobile Terminal 101
4.3 SPECIFICATION OF THE PROCEDURES 102
4.3.1 Location Update Registration: initial and “off-call” mobility management 103
4.3.2 User Registration 104
4.3.3 Session Establishment 105
4.3.4 On-Call Mobility: the handover procedure 107
4.3.5 Comparison with canonical SIP based mobility 108
4.4 IMPLEMENTATION AND MEASUREMENTS 108
4.4.1 Evaluation of access network performance 110
4.4.2 Evaluation of handover performance 111
4.5 RELATED WORK 116
4.6 CONCLUSIONS 117
5 LINK LAYER: MULTIPLE-PATH ROUTING AND LOAD BALANCING APPROACH FOR WIMN 119
5.1 INTRODUCTION 119
5.2 MOTIVATIONS 119
5.3 APPROACH 120
5.3.1 Multiple Path creation 120
5.3.2 Dynamic Path Selection 121
5.3.2.1 Path quality measurements 122
5.3.2.2 Path selection decision 122
5.4 CONCLUSIONS 123
References 124
Appendix I Analysis with different versions of TCP 129
Appendix II Rate Estimation Algorithm 131
Appendix III Assumptions and model limits 132
Appendix IV Alternative computation of the system in lossless AP case 134
Appendix V Proof of the effectiveness of the VBN architecture 136

1 Transport layer: TCP Fairness Issues - Problem Analysis and solutions Based on Rate Control

1.1 Introduction

In this Chapter, 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.

1.1.1 Motivations and Problem Statement

In the considered 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 it is 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, the 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.).

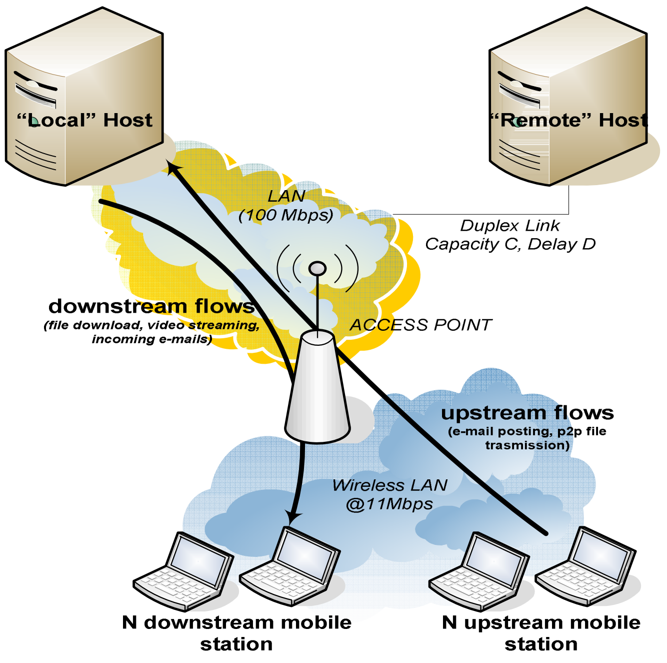
"Local" Host
"Remote" Host
LAN
(100 Mbps)
Duplex Link
Capacity C, Delay D
downstream flows
(file download, video streaming, incoming e-mails)
ACCESS POINT
upstream flows
(e-mail posting, p2p file trasmission)
Wireless LAN
@11Mbps
N downstream mobile station
N upstream mobile station

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. 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 [18] because of the arising congestion and corresponding packet losses happening in the download buffer at the Access Point [19]. 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 [7]. 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.

1.1.2 Related Work and Basic Assumptions

Several papers proposed solutions to alleviate unfairness phenomena. The proposed solutions can be classified according to the layer at which they operate (MAC [13], IP [14][15], TCP [3][16]) or according to the level of fairness that they achieve (per-connection fairness [16][17][15], aggregate upstream/downstream fairness [14], per-station fairness [3][13], etc.).

In [3] 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 1.3). Unfairness among TCP and UDP flows in more complex topologies (i.e., with multiple WLANs) has been preliminary discussed in [4].

In this section, 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. The 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) [5]. 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 Chapter, in Section 1.2 we discuss the “basic” system model without rate control mechanisms and introduce, evaluate and comment some performance measures. In Section 1.3, we present our solutions based on rate control, to face the so called critical unfairness. The static approach is detailed in Section 1.4, together with the related performance evaluation; similarly, we present the adaptive rate control in Section 1.5 while the related performance evaluation (also in presence of TCP short-lived connections) is presented in Section 1.6. Finally Section 1.7 proposes the performance analysis of our solution in a real test-bed.

1.2 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 work, 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 scenario is shown in Figure 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 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) [6]. 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 [8]. 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 [8] and SACK [9], to verify that the phenomena under study are not specifically tied to the selected TCP version (see Appendix I).

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 Section 1.6; the greedy sources traffic scenario is the main one, described and analyzed through this Section.

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 [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[1]. 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 $R_{\mathrm{up\_tot}}$ (i.e., the sum of the throughputs of upstream TCP connections), the total

[1] 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.

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.

1.2.1 Numerical results

We start our analysis with the case of "no rate-control", that is without any special mechanism 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 600 seconds of simulated time; the throughput was measured during the last 200 seconds of each experiment, to avoid transient effects and operate in steady state conditions.

Figure 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 ($N=N_{dn}=N_{up}$), when no rate control is implemented. 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 buffer.

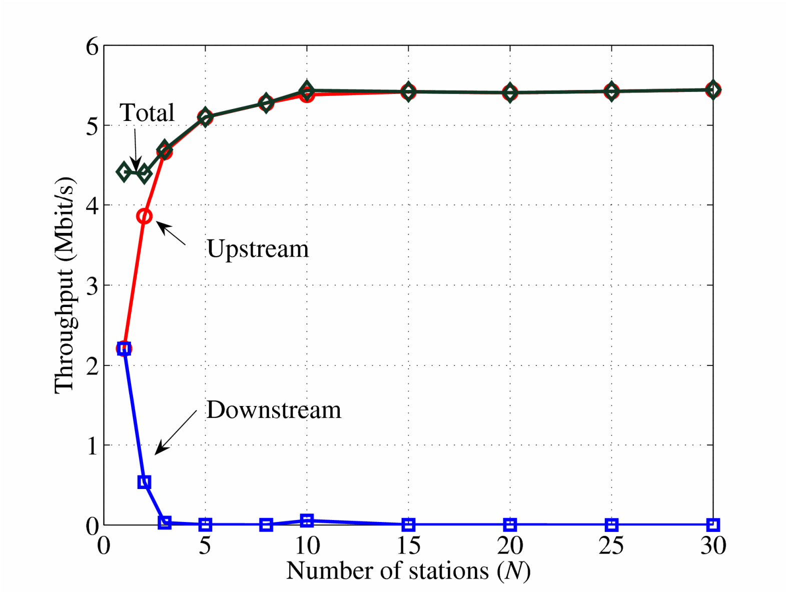

Figure 2 - Upstream, downstream and average total throughput ("local wired host" scenario) without rate control mechanisms

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[2]. 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).

We focus now on upstream connections. Figure 3 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 4, it sharply increases for N>5.

[2] If the "delayed ACK" mechanism is used, there will be one ACK for every two TCP segments, in the no loss case, but the effect of the loss is the same.

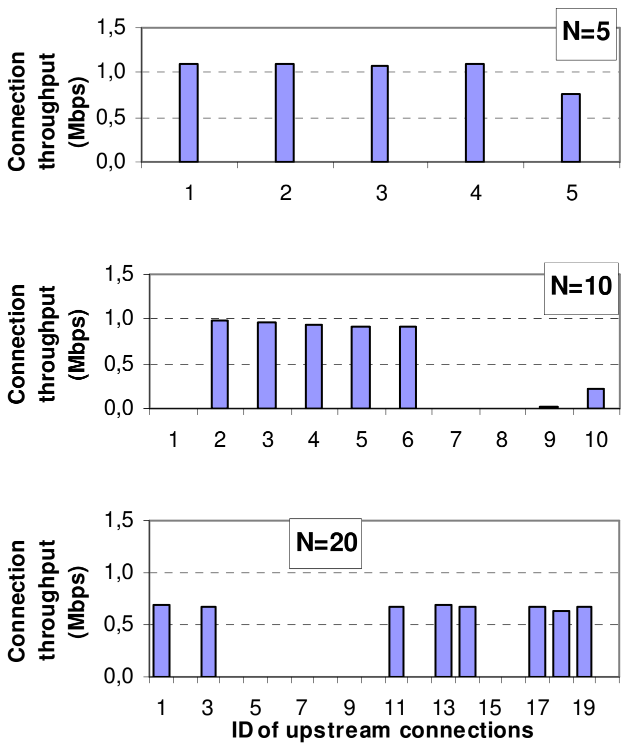

Figure 3 - Throughput of upstream connections ("local wired host scenario")

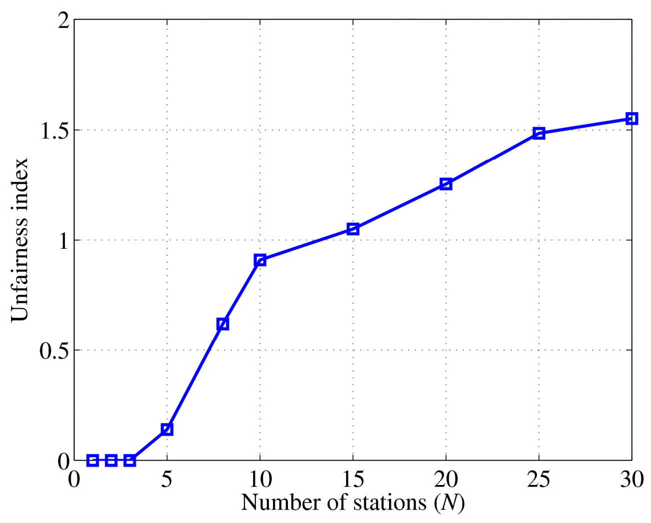

Figure 4 - Ratio σup/Rup for upstream connections ("local wired host scenario")

1.2.2 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, as shown in Figure 2.

To support this hypothesis, we evaluated by simulations (see Figure 7, discussed below) the packet loss probability in the downlink buffer of the Access Point as a function of N. The main results are the following. 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., in Figure 7, discussed below we show 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 [7]. 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 3. 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 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·*N*·CW_max, where CW_max is the TCP maximum congestion window size (expressed in packets of 1500 bytes, following NS-2 conventions).

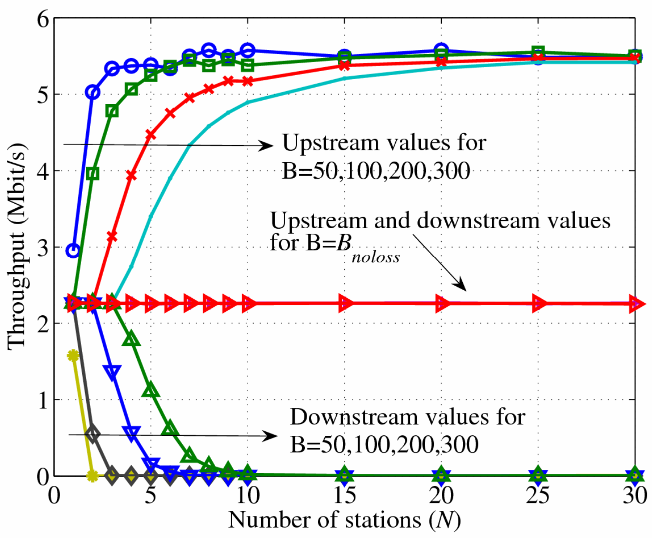

Figure 5 - Total upstream and total downstream throughput

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 5). 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). 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 6, where we plot the throughput of individual connections.

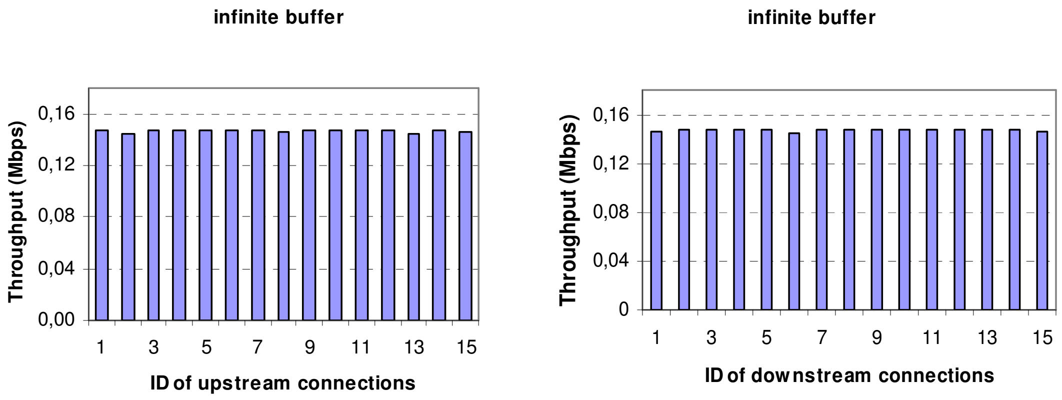

Figure 6 - Fairness achieved in lossless conditions

To verify our conjecture about the "Stop&Wait" behavior, we have complemented our throughput measurements (shown in Figure 5) 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).

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 [3]. 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. 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 5 were used for Figure 7 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.

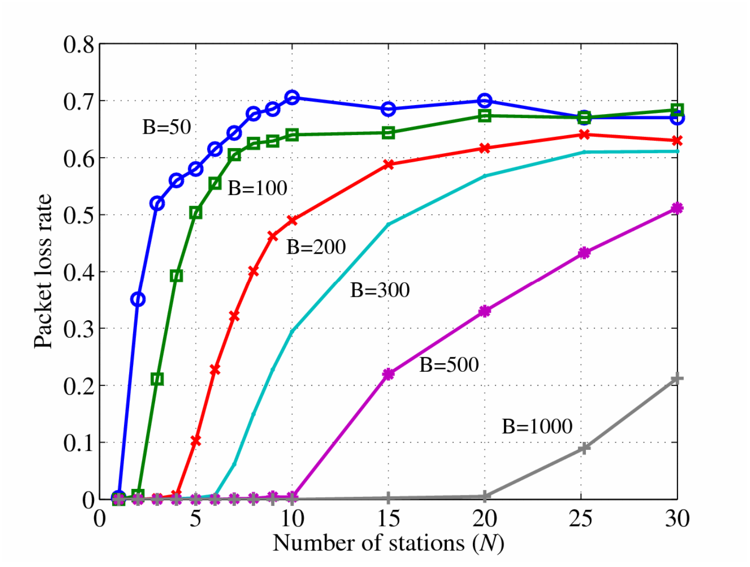

Figure 7 - Packet loss rate in the downlink Access Point buffer ("local wired host" scenario)

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 2). 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 2 in Section 1.2.1).

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 2).

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 2). 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. This is shown to be actually true in the Appendix I for the cases of TCP SACK and TCP NewReno.
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.

1.3 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 happening in the Access Point). In this case, fairness conditions are achieved since, as discussed in Section 0, each source operates in a "stop & wait" mode[3].

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.

[3] 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 Figure 5, when the buffer size is equal to B_{noloss}.

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 introduces an additional loss, by randomly dropping packets, even when there would be enough tokens in the bucket to forward packets, 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 $0<T_{\text{h}}<1$ be the threshold level at which the STBF should start dropping packets. The STBF introduces a loss probability $P_{\text{drop}}(H)$ for an incoming packet as follows:

$$P_{drop}(H) = \begin{cases} 0 & \text{if } H > Th \cdot B_{bucket} \\ \\ \dfrac{Th \cdot B_{bucket} - H}{Th \cdot B_{bucket}} & \text{if } H < Th \cdot B_{bucket} \end{cases}$$

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 1.4 and 1.5 respectively.

1.4 Static Rate Control

The Figure 8 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. 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 intersting test-bed, useful to make practical experiments. Such solution is depicted in Figure 9.

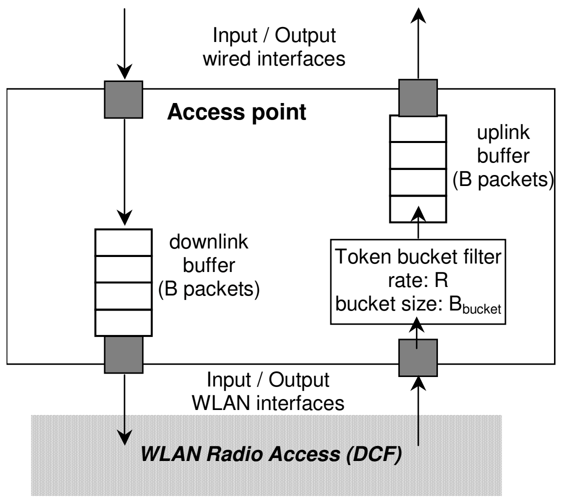

Figure 8 - Rate control solution based on a rate-limiter

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. The final choice is *R*=2.3 Mb/s, B_{bucket}= 500 packets of 1500 bytes and T_h=*0.9*.

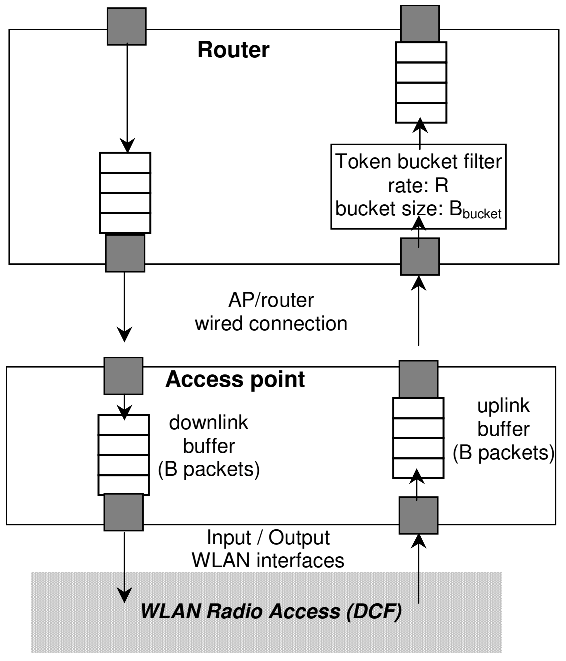

Figure 9 - Rate-limiter placed on an external device

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.

1.4.1 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 1.2, 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 Figure 10, 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.

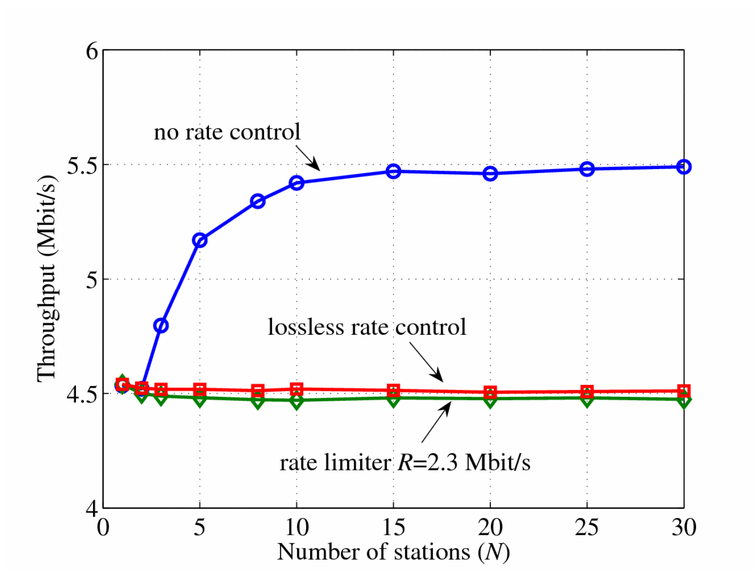

Figure 10 - Average total IP level throughput

The total upstream and the total downstream throughput are reported in Figure 11. 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. 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.

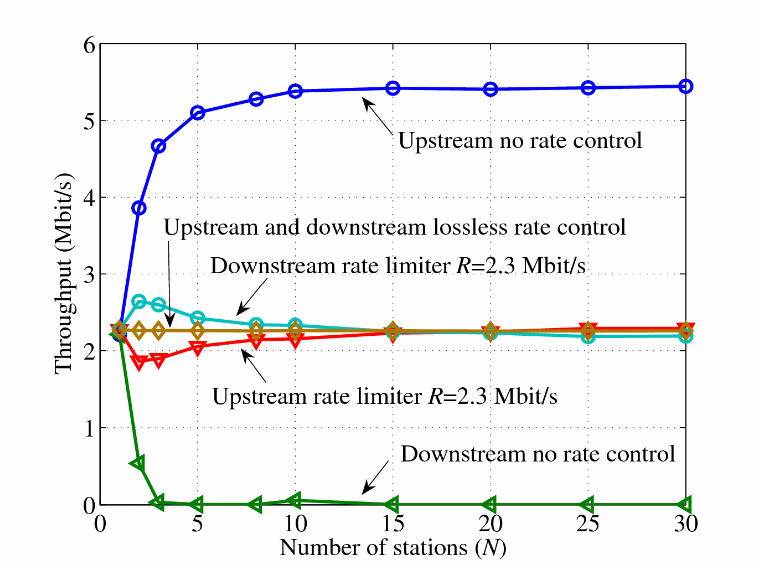

Figure 11 - Upstream and downstream throughput

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 Section 1.4.1.1) 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 [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 sub-Section we highlight the limitations of this static approach and propose an adaptive mechanism.

1.4.1.1 Validation of the Static Rate Limiter for Different RTT

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 [2]. 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. Note that this analysis is performed using the "classical" TBF (not the smoothed one - STBF).

Figure 12 shows 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 [3], and our 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.

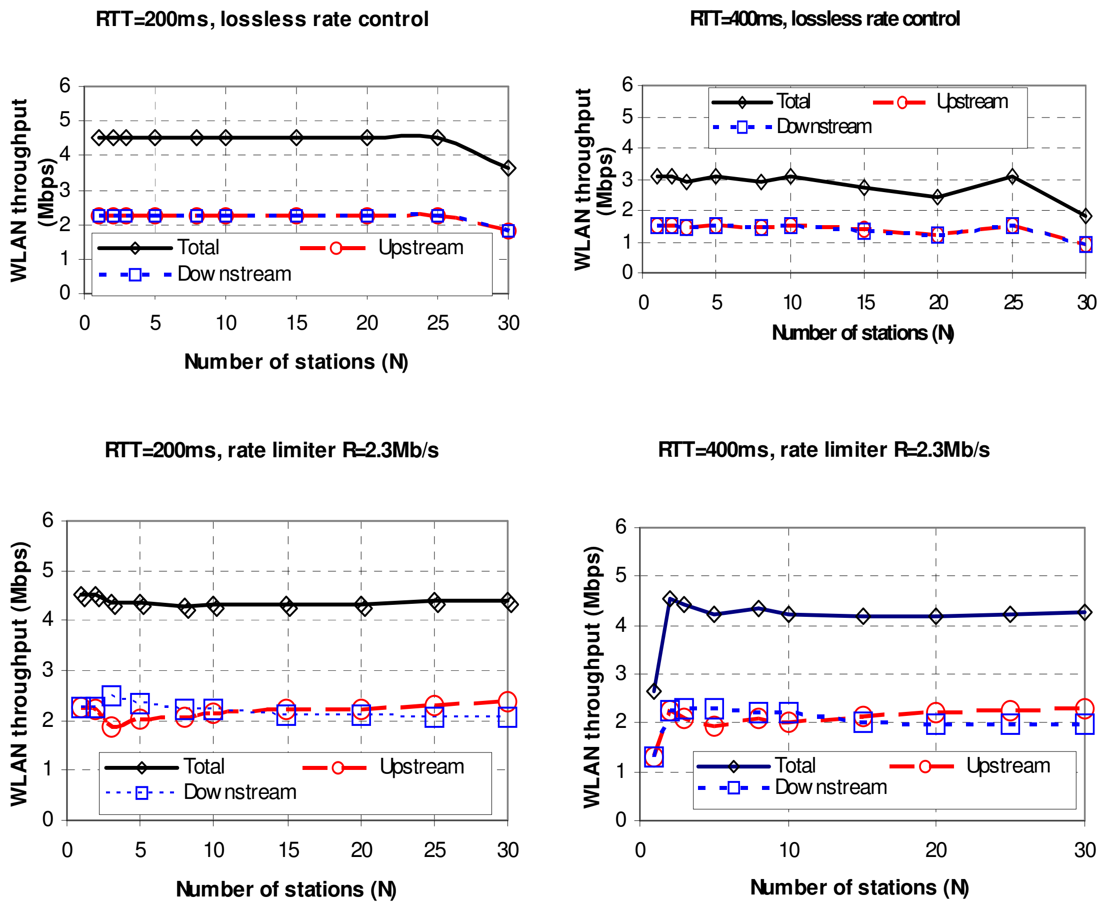

Figure 12 - Throughput versus N in the "remote wired host" scenario (RTT=200, 400); comparison between lossless rate control and static rate limiter

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. The well know throughput limit of the window based protocols (R≤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 13 provides another insight on the RTT problem, as it reports the total throughput versus RTT for the "no rate control", lossless rate control, and rate limiter solutions for a fixed number of connections (N=15). The throughput in case of lossless rate control decreases starting from RTT≅250 ms.

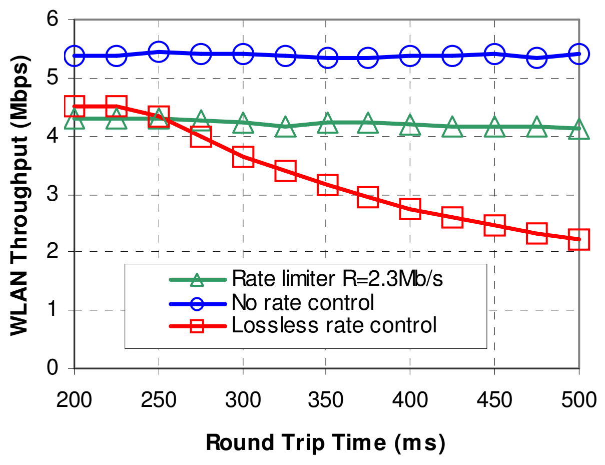

Figure 13 - Total throughput vs. RTT, N=15 ("remote wired host" scenario)

1.5 Adaptive Rate Control

The static mechanism described in the previous Section has two major problems: i) 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; ii) 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 1Mb/s to 11Mb/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 $R_{\text{down}} < C\text{-}R$, then the static rate limiter causes a waste of capacity in the order of $C\text{-}R\text{-}R_{\text{down}}$. The idea of the adaptive rate control is to increase the rate of the token bucket filter in these conditions up to $R'=C\text{-}R_{\text{down}}$ 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 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 1.2, 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[4]. We denote by C_{max} the total estimated throughput (i.e., $C_{\text{max}} = R_{\text{up}} + R_{\text{down}}$). 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 $T=kT_{\text{p}}$, can be expressed as:

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

```
$R_{up}$:   estimated throughput from the WLAN interface to the AP at time
k·$T_p$;
$R_{down}$: the estimated throughput from the wired interface to the AP at
time k·$T_p$;
NL :  number of packets lost at the downlink queue in the time [(k-
1)$T_p$, k $T_p$ )
-   at time k·$T_p$ 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);
```

In case of loss, the bucket rate is decreased down to the target rate, set as one half of the estimated current capacity. When there is the first loss event after a sequence of time intervals without loss in the downlink buffer, the rate is set to the current evaluated rate in the upstream direction R_{up}, minus R_{step} and the number of tokens in the Token Bucket Filter is set to the threshold value where it can start dropping packets.

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. 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.

1.5.1 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.

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

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 2.

Parameter		**Value**
AP downlink buffer B	(packets)	100
TBF bucket size B_{bucket}	(bytes)	500*1500
Th		0.9
R_{step}	(kbps)	200
T_p	(ms)	300 ms

Table 2 - Adaptive rate limiter parameters

Figure 14 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).

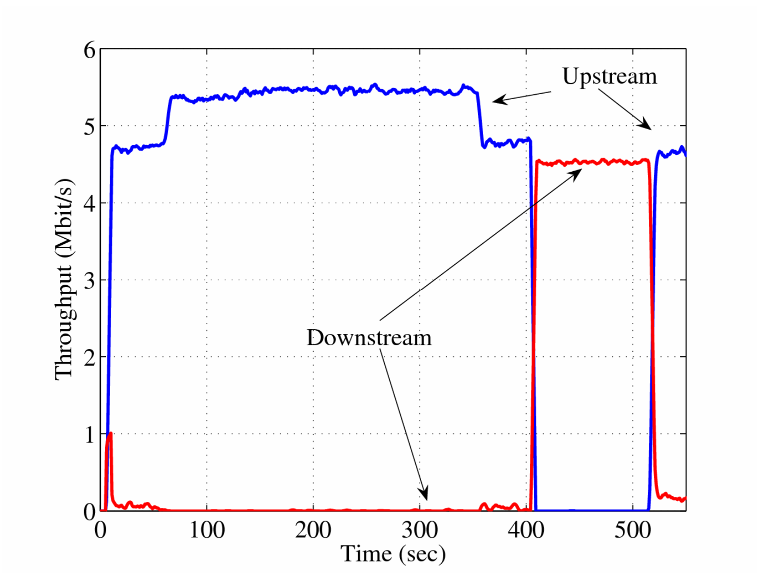

Figure 14 - Time evolution of upstream and downstream throughput without rate control

Figure 15 reports the temporal evolution of the throughput obtained by using the static rate limiter. We note 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. 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.

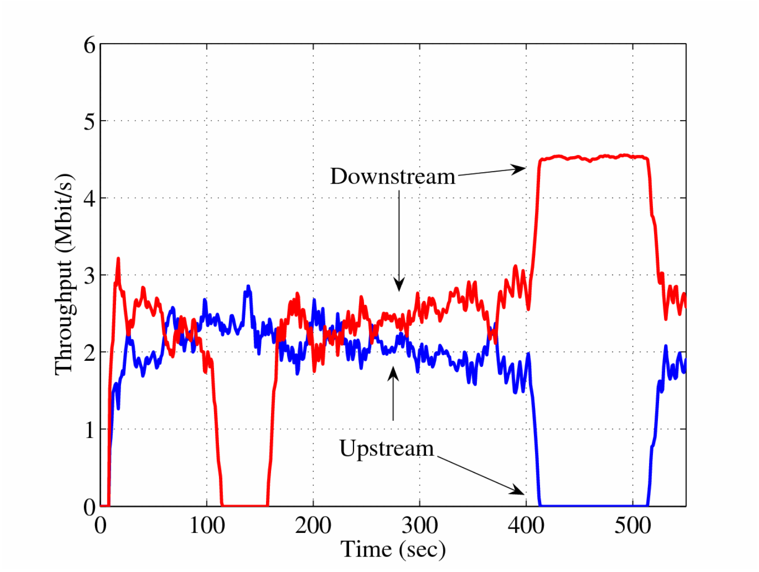

Figure 15 - Time evolution of upstream and downstream throughput with static rate control

Finally, Figure 16 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.

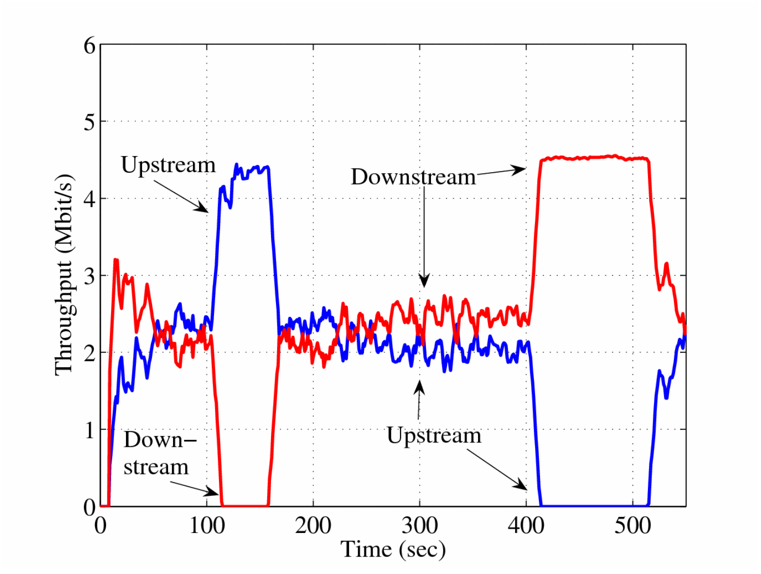

Figure 16 - Time evolution of upstream and downstream throughput with adaptive rate control

1.6 Analysis with short-lived sources

In this Section we analyze the fairness performance of our system in two scenarios: i) system loaded with short-lived sources; ii) system loaded with a mix of short-lived and greedy sources. We recall that, in our scenario, unfairness phenomena are due to losses occurring in the downlink buffer. Greedy up-stream connections produce a heavy loading of the downlink buffer, for long periods of time. As a consequence, downstream connections perceive high segment losses, severely limiting their performance with respect to the upstream connections, which mainly experience ack losses. In these conditions, critical unfairness shows up.

On the other side, short-lived upstream connections generate a lighter load of the downlink buffer since their TCP congestion windows do not have the time to reach large values. As a consequence, competing downstream connections succeed in better accessing buffer resources and enjoying better fairness performance than in the previous case. In these conditions, critical unfairness does not occur (at least in our scenario; it is clear that we are talking of a continuous process and that by increasing the source duration, sooner or later we will reach critical unfairness conditions; the definition of short-lived sources is rather a qualitative one).

1.6.1 Short-lived TCP sources

Short-lived sources model the download or the upload of a small amount of data (in our setting 100 kbytes). We considered a dynamic scenario in which we activate 120 upstream connections and 120 downstream connections in the time interval lasting from t_0=20s to t_1=50 seconds; connections are activated one after the other, every 30/240 s, alternating an upstream one and a downstream one. Connections remain active until they transfer all their data. The system is loaded so that the WLAN capacity is saturated and the system works in conditions similar to the greedy sources scenario.

Figure 17 shows the aggregate throughput of upstream and downstream connections and the overall throughput, without rate control. We can see that: i) the overall throughput reaches the system capacity; ii) data transmission ends at time t_2=78 seconds showing that all connections succeed in completing their transmission; iii) the system reaches a good level of fairness between upstream and downstream connections, with a slight prevalence of the former ones (curves cross in the last part of the simulations only because upstream connections are ending their transmission before than the downstream ones and thus leave space for the latter ones); iv) critical unfairness never occurs.

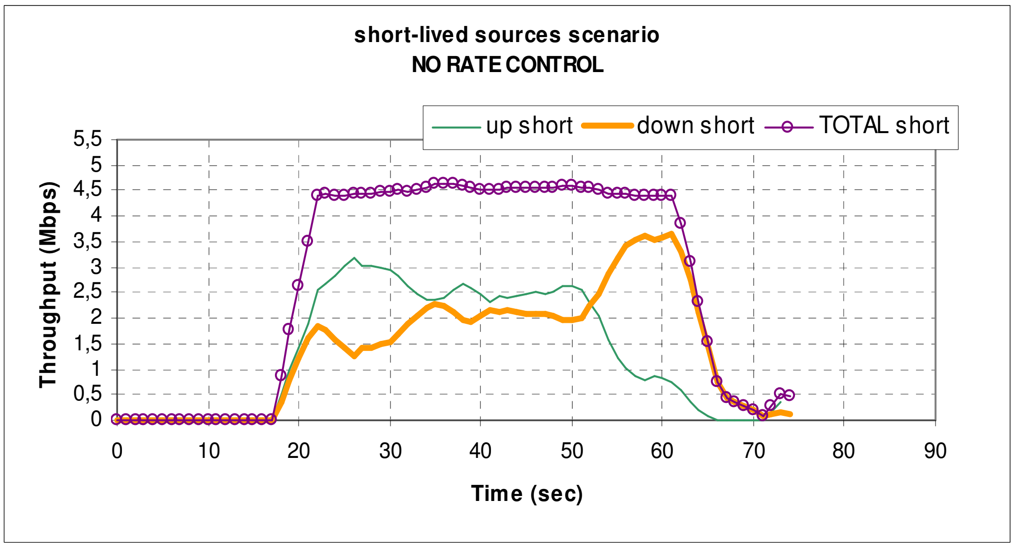

Figure 17 – Aggregate throughput of upstream and downstream connections (short-lived sources scenario)

Figure 18 shows the same setting but with our adaptive, dynamic rate control in action: our mechanism has a limited effect since the system is already operating in good fairness conditions; the mechanism slightly improves fairness between upstream and downstream but is not able to reach a perfect sharing. The reason is that the greater traffic burstiness of short-lived connections yields an over estimation of the parameter $C_{\max}$ (defined in Section 1.5) and thus limits the upstream traffic to a greater value (i.e., 2.5 Mbit/s) than the optimal one (i.e., 2.3 Mbit/s).

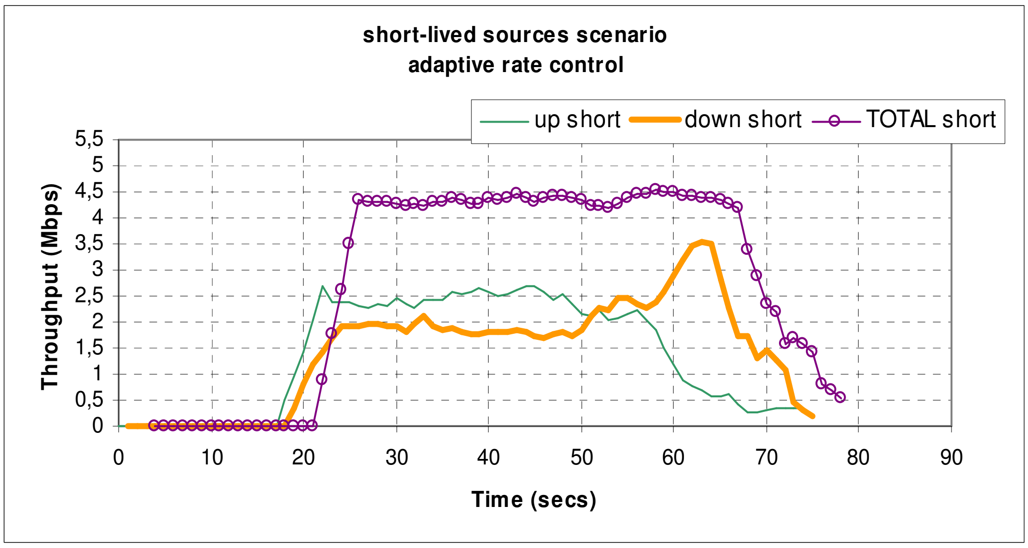

Figure 18 - Aggregate throughput of upstream and downstream connections (short-lived sources scenario), with adaptive rate control

1.6.2 Mix of short-lived and greedy sources

This traffic scenario comprises the same short-lived connections of the previous case, activated in the same way, plus three greedy upstream connections and three greedy downstream connections activated at time t0=0.

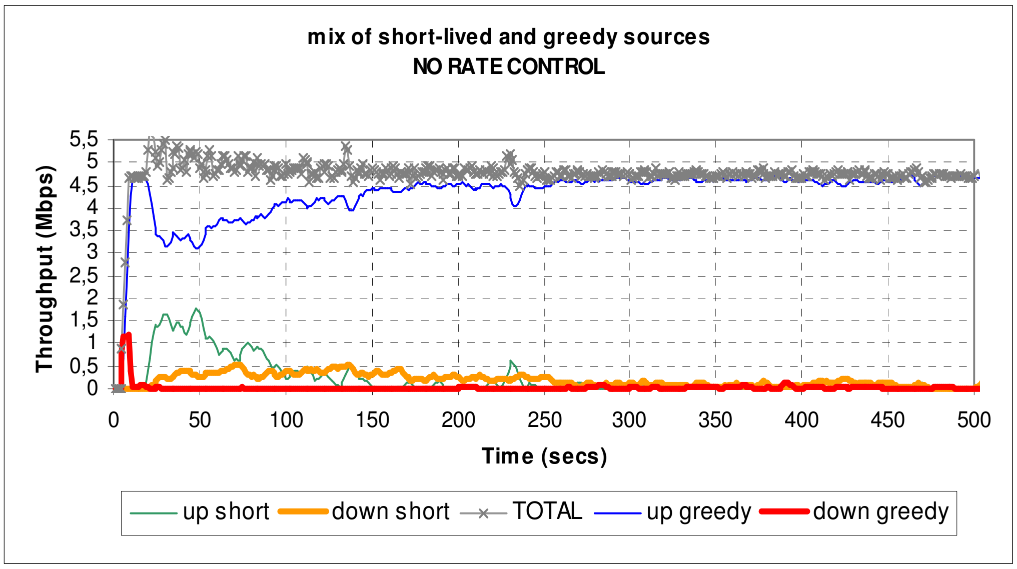

Figure 19 - Aggregate throughput of upstream and downstream connections (mix of short-lived and greedy sources)

Figure 19 shows the aggregate throughput of both short-lived and greedy upstream and downstream connections plus the overall throughput, without rate control. We can see that: i) greedy upstream connections cause the critical starvation of greedy downstream connections; ii) short-lived connections are heavily penalized: their throughput is much less than that of the greedy upstream connections; we also stress that we are speaking of 240 short-lived connections and thus the throughput perceived by the single connection can be very small or even zero.

Figure 20 shows the same setting but with our rate control in action. The system should be observed when all sources are active and have reached a stable state, i.e., in the time interval lasting from t0=30s to t1=40 seconds. The effect is: i) greedy downstream connections perceive the same throughput of the greedy upstream ones; ii) short-lived traffic gets now a greater share of system resources; iii) overall, capacity is fairly shared among the four classes; we point out that although the curves show that the overall short-lived throughput is greater than the greedy one, we must remember that we have 240 short-lived connections against 6 greedy connections. If we take into account this, we find that the capacity is indeed fairly shared among the four classes. We can conclude by saying that our mechanism is effective in all conditions, even if, when the system is loaded with short-lived traffic only, it is not really necessary.

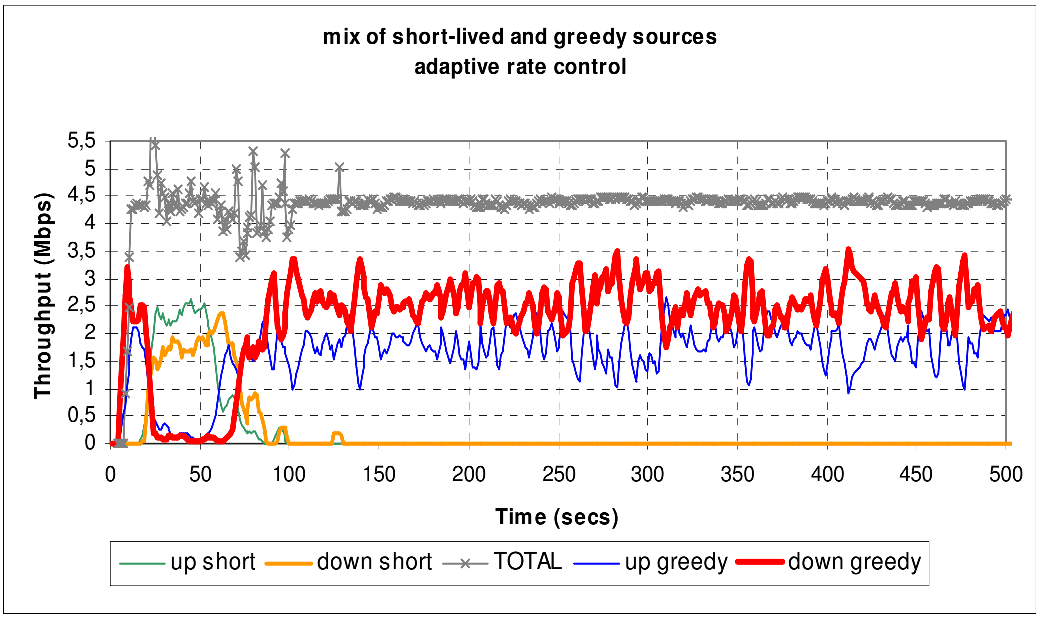

Figure 20 - Aggregate throughput of upstream and downstream connections (mix of short-lived and greedy sources), with adaptive rate control

1.7 Performance evaluation in a real test-bed

In the this Section (instead of resorting to simulations like in the previous Sections) the performance analysis is performed in a real test-bed and proves the feasibility and effectiveness of the proposed mechanism. To allow fellow researchers to reproduce this work we published on the WEB all the implementation code. Again, to the best of our knowledge, this is the first example of a mechanism to provide fairness in a WLAN that it is implemented and tested in the field.

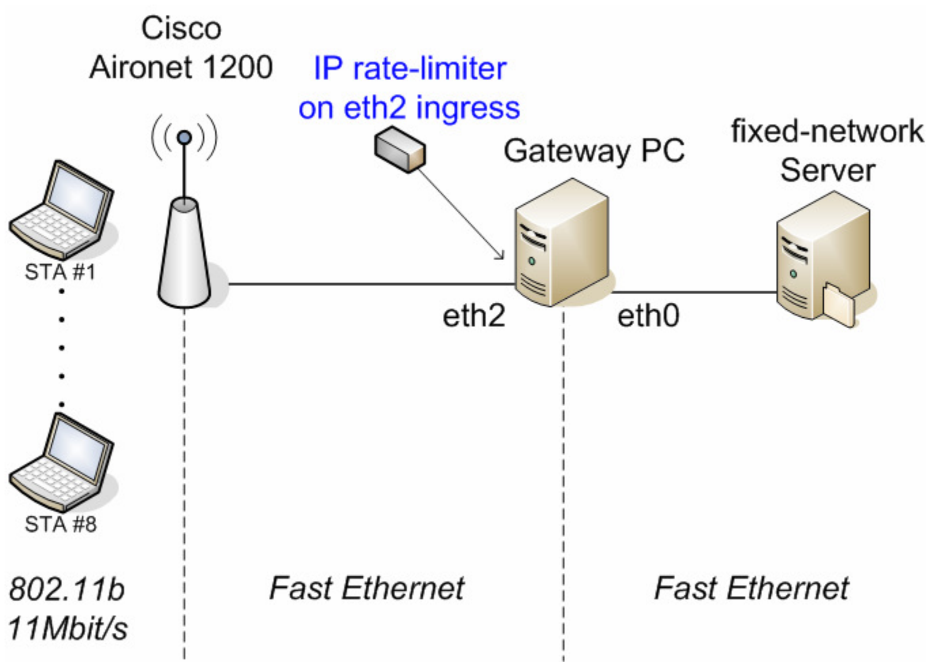

Figure 21 – Test-bed layout

Figure 21 depicts the test-bed layout, while Figure 22 shows a picture of the test-bed.

The test-bed includes ten Personal Computers (PCs) and an access point Cisco Aironet 1200. A PC plays the role of the server residing in the fixed network. Another PC is the network gateway. The remaining 8 PCs are the STAs, equipped with 802.11b cards at 11Mbit/s. All the PCs mount the Microsoft Windows XP operating system, with the exception of the network gateway, running Linux, kernel version 2.6.17.13, Kubuntu distribution.

The gateway has two Fast Ethernet interfaces, named *eth0* and *eth2*. The *eth0* interface connects the gateway to the fixed-network server, while *eth2* connects the gateway to the AP. As regards the addressing space, the gateway routes between two different networks: 192.168.100.0/24 on the *eth0* side; and 10.0.0.0/8 on the *eth2* side.

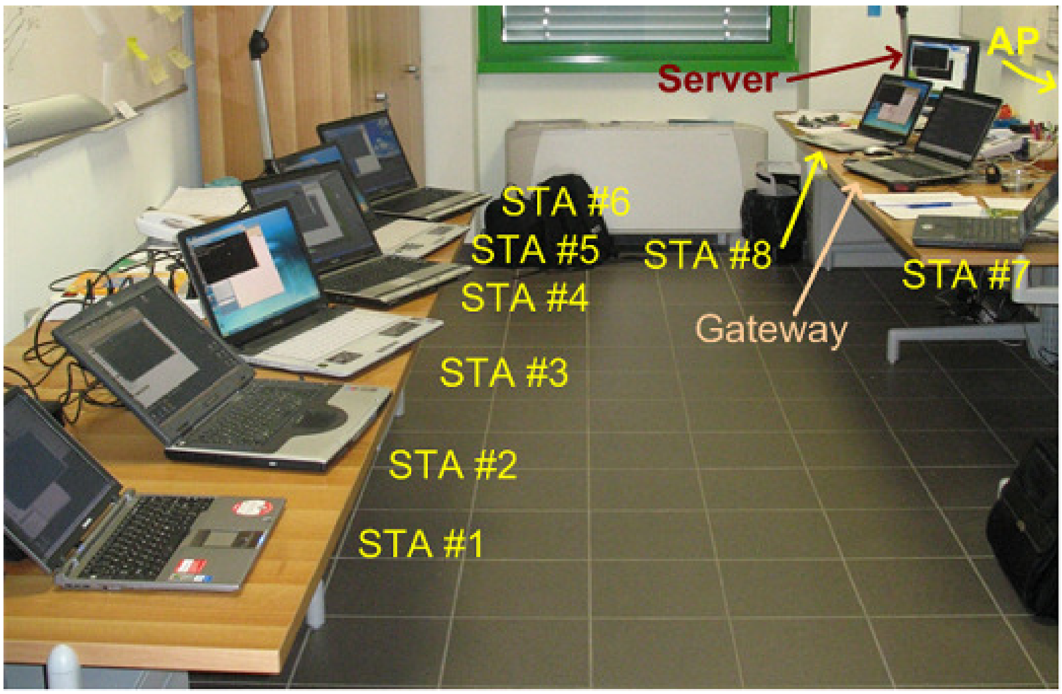

Figure 22 – The test-bed

The IP rate-limiter with adaptive rate control is located within the gateway and operates at the ingress of the *eth2* interface. It has two components: i) the TBF policer built-in in the Linux kernel that acts as the IP rate-limiter; ii) the user-space application *rl* that implements the computation of the dynamic value of *R* as described in Section 1.5. The original TBF policer of the Linux kernel has been modified to allow a user-space application to vary at run time the TBF rate. The *rl* application has been developed from scratch. All the software code is available in [11].

We evaluated the system performance with two different TCP traffic models: static and dynamic. The first one allows to assess the performance in steady state, while the dynamic traffic model allows to highlight what happens when traffic conditions change.

In the case of *static traffic model*, we assume that a single STA supports one downstream and one upstream TCP connection with the fixed-network server. As a consequence, if we let N the number of STAs, then the number N_{up} of upstream connections and the number N_{down} of downstream connections are equal to N. Each TCP connection is active for the entire duration of the test, that is 5 minutes.

In the case of *dynamic traffic model*, we use 6 STAs and the number N_{up} of upstream and N_{down} of downstream active connections versus the time is reported in Table 3.

N_{up}	6	6	6	0	6
N_{down}	6	0	6	6	6
Time (s)	20÷100	100÷200	200÷300	300÷4000	400÷500

Table 3 - Number of connections versus time in case of dynamic traffic model

The main TCP parameters are: segment payload=1460 bytes; max congestion window=65536 bytes. We repeat the tests two times: with and without IP rate-limiter with adaptive rate control.

The merit figures that we consider are the upstream, downstream and total (i.e., the sum of these two components) TCP goodput. The TCP goodput is defined as the number of useful bits per unit of time received at the connection sink.[5]

1.7.1 Test-bed results without rate-limiter

Figure 23 reports the average upstream, downstream and total goodput as a function of the number *N* of STAs in the case of static traffic model. The total goodput that we measure is in line with the theoretical value that one could expect, considering the fact that the TCP implementation of Microsoft Windows uses the delayed ACK policy and sends a TCP ACK each two received TCP segments.

For all the values of *N* reported in the figure, the downstream goodput is significantly lower than the upstream goodput. This is an evidence of the upstream/downstream unfairness. A critical unfairness occurs for *N* greater than 4 STAs.

Figure 24 shows the average upstream and downstream goodput for single STAs with *N*=6 and a static traffic model. We observe also an intra-upstream unfairness, even though not a critical one. For instance, STA #4 perceives an upstream goodput equal to about two times the upstream goodput of STA #1.

[5] The measurement campaign was not performed at the same time of the simulation study and for these measures we choose as merit figure the TCP goodput (while in the simulations we considered the TCP goodput). With respect to the TCP throughput we need to take into account the absence of the IP header and the TCP ACKs

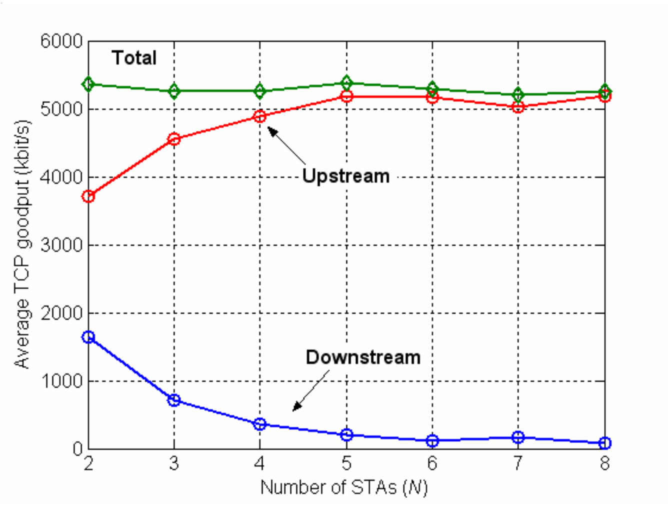

Figure 23 - Average upstream, downstream and total TCP goodput as a function of N (static traffic, without rate-limiter)

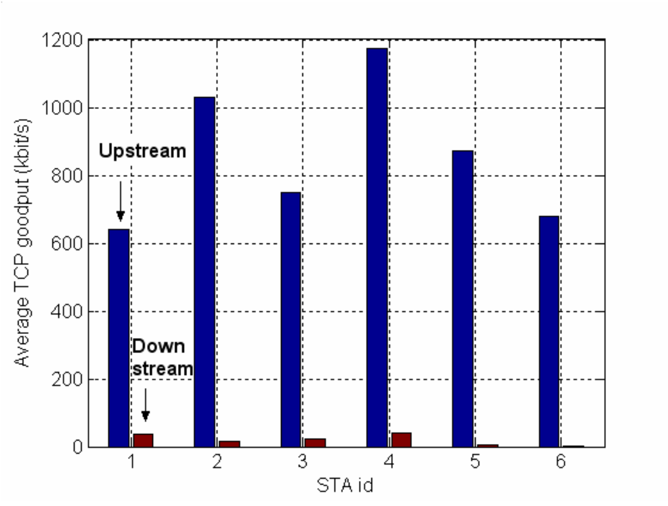

Figure 24 - Average upstream and downstream TCP goodput for single STA (N=6, static traffic, without rate-limiter)

In the previous Sections we showed by means of simulations that also the intra-upstream unfairness may become critical, in the sense that some upstream connections are unable to start. This phenomenon occurs when a large number of STAs are present, thus, our test bed does not reproduce this behavior.

Figure 25 reports the instantaneous upstream, downstream and total TCP goodput for *N*=6 and with a static traffic model. Most of the total goodput is made up of upstream goodput, while the downstream connections are almost starved.

Finally, Figure 26 shows the instantaneous upstream, downstream and total TCP goodput with a dynamic traffic model. We can see that the downstream traffic survives only when there are not upstream connections, i.e., between time interval 300-400 s.

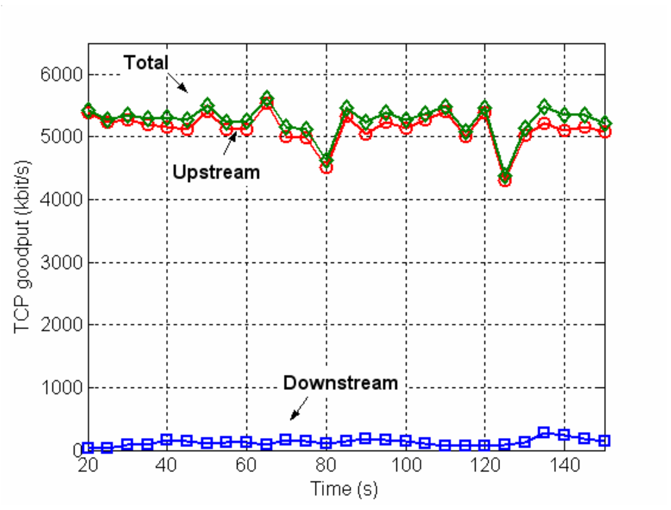

Figure 25 - Instantaneous upstream, downstream and total TCP goodput (N=6, static traffic, without rate-limiter)

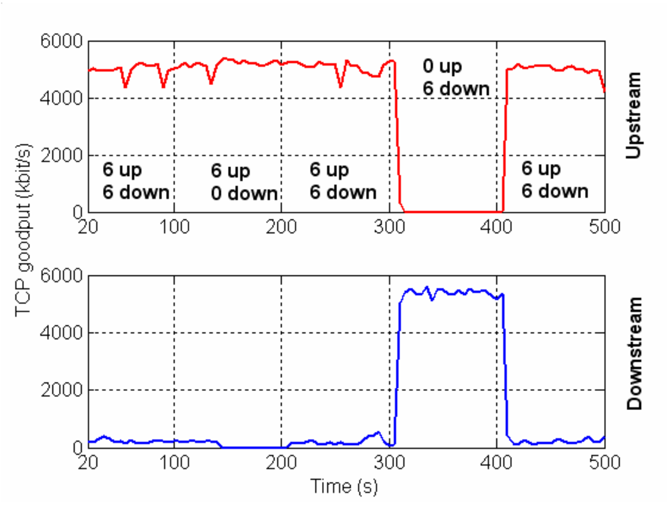

Figure 26 - Instantaneous upstream, downstream and total TCP goodput (dynamic traffic, without rate-limiter)

1.7.2 Test-bed results with rate-limiter

In these tests we assume that: i) the adaptive rate control refresh period T_p is equal to 1 s; ii) the rate step R_{step} is equal to 200 kbit/s; iii) the bucket size B_{bucket} is equal to 250 kbytes. The latter value has been chosen by means of test trials, maximizing the fairness for the numbers of STAs considered in this Section. However, as noted above, different choices in the range of values that we examined (50-500 kbytes) would have had a limited impact on performance.

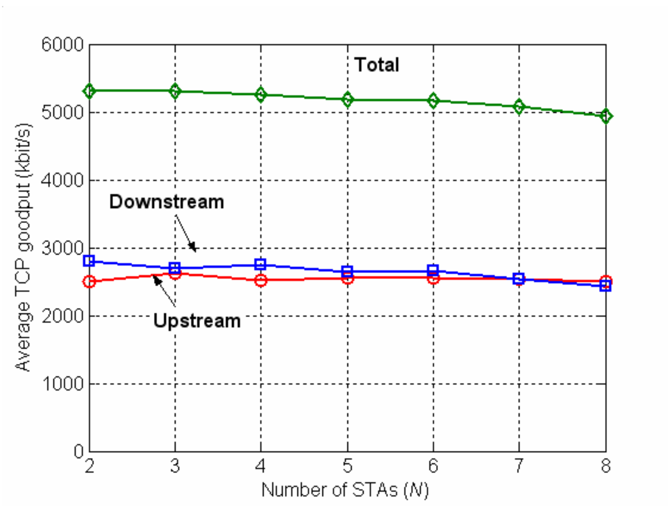

Figure 27 - Average upstream, downstream and total TCP goodput as a function of N (static traffic, with rate-limiter)

Figure 27 reports the average upstream, downstream and total TCP goodput as a function of the number *N* of STAs with a static traffic model. The total goodput is very similar to that plotted in Figure 23, i.e., the one obtained without rate-limiter. Thus, the IP rate-limiter does not waste radio resources. Moreover, the upstream and downstream goodputs are almost equal to each other, confirming the effectiveness of our approach. Figure 28 shows the average upstream and downstream TCP goodput for single STAs with *N*=6 and a static traffic model. It is interesting to note that we obtain also an intra-upstream and intra-downstream fairness.

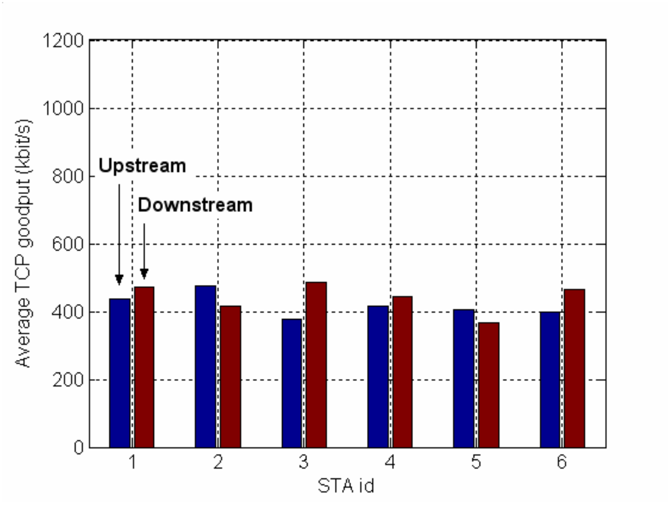

Figure 28 - Average upstream and downstream TCP goodput for single STA (N=6, static traffic, with rate-limiter)

Figure 29 reports the instantaneous upstream, downstream and total TCP goodput with *N*=6 and a static traffic model. We note that upstream, downstream goodput are almost equal to each other not only in average, as shown in Figure 27, but also in the short term.

Finally, Figure 30 shows the instantaneous upstream, downstream and total TCP goodput with a dynamic traffic model. We can see that when both the upstream and downstream connections are active, the radio resource are fairly shared, since the upstream and downstream goodput are almost equal to each other. Moreover, when the upstream or the downstream connections are switched off, the remaining active connections use all the radio bandwidth, without wasting radio resource.

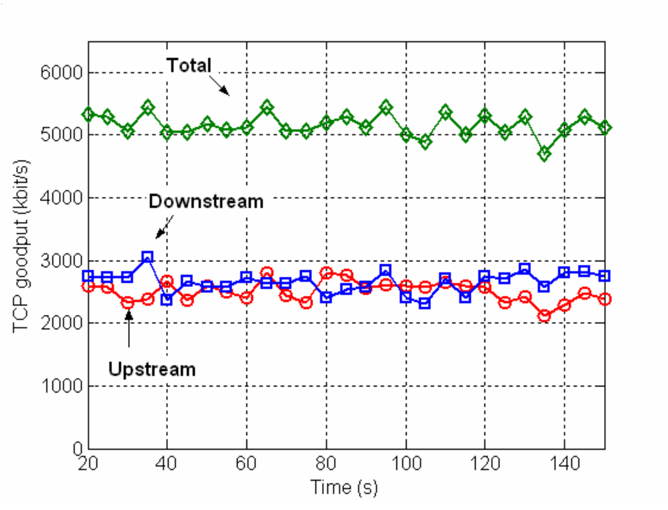

Figure 29 - Instantaneous upstream, downstream and total TCP goodput (N=6, static traffic, with rate-limiter)

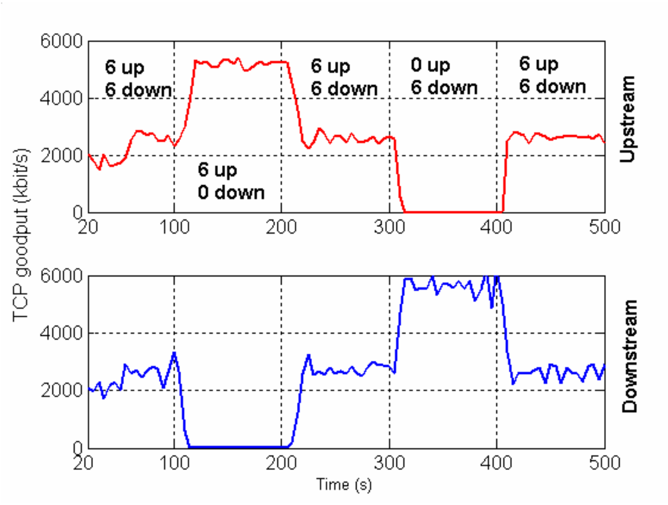

Figure 30 - Instantaneous upstream, downstream and total TCP goodput (dynamic traffic, without rate-limiter)

1.8 Conclusions

In this Chapter, we addressed fairness issues in a wireless access network based on the IEEE 802.11b standard, operating in DCF mode at 11 Mbps. 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.*

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.

2 Transport Layer: Analytical Model of TCP Fairness and Customizable Framework for Traffic Control

2.1 Introduction

As illustrated in the previous Chapter, the *upstream-downstream* unfairness derives from the fact that although the AP must support the majority of traffic, the 802.11 MAC layer assigns the same transmission opportunities of a single STA to the AP. This leads the AP to be the network bottleneck; i.e., to become an accumulation point for TCP in-fly packets. Since the upstream TCP connections are practically insensible to packet losses occurring in the AP, the upstream connections fill the space in the AP buffer with their TCP ACKs and lead the AP buffer to work under heavy loss condition[6]. In these loss conditions, the downstream TCP connections are starved by the heavy segment loss that they experience in the AP buffer.

The upstream-downstream unfairness theoretically appears as critical problem that may lead the customers to a strong dissatisfaction. Nevertheless, in our experience[7] the customers of Wi-Fi hot-spots do not show such a dissatisfaction since they more or less succeed in transfer data. This means that in practical scenarios, it is quite rare that the AP works in a heavy loss.

In our opinion, what up to now has hidden the unfairness issue to the customers is the fact that the most widespread Operative System (OS) on laptops was Microsoft Windows XP, which sets the default value of the socket buffer (both sender and receiver side) to 8 kB. This implies that for a single TCP connection, there are at most 6 in-fly packets. As the AP buffer space of the commercial equipments is in the order of 100 packets, the small number of in-fly packets practically prevents the AP from heavy packet losses. Consequently, the number of TCP flows that should be active at the same time to provide heavy packet loss in the AP buffer is in the order of twenty connections.

[6] We note that an enough number of upstream connections is required in order to fill the AP buffer with TCP ACKs. The number of required upstream connections strongly depends on the management strategy of the AP buffer: packet-level or byte-level. In the former strategy a small number of upstream connections may be enough to fill the AP buffer (case under study in this Chapter). On the contrary, in case of byte-level management of the AP buffer the small size of the TCP ACK requires a very great number of upstream TCP connections representing a scenario with limited practical interest.

[7] Based on the Wi-Fi hot-spot deployed in our University and used by students and personnel.

While the previous discussion has depicted why up to now the upstream-downstream unfairness issue is not showed up in a practical scenario, we need to observe that the novel Microsoft OS, i.e. Vista [8], uses an automatic tuning [70][71] mechanism that significantly increases the value of the socket buffer. Vista uses a socket buffer of 65 kB as default value; but it may also use a greater value when the device at other end of connection supports the TCP window scaling option. This means that a small number of upstream TCP connections (e.g., five) is enough to produce heavy packet loss in the AP and to show up the unfairness issue.

Considering that the current laptops' market is widely adopting Microsoft Vista as preloaded OS, we think that even if the unfairness problem was not realistic yesterday, it may become dramatically realistic tomorrow; at least for those scenarios where the concurrence of long uploads is not a rare event. For instance, we may think to an Hot-Spot used by sportive journalists that send their pictures to the editorial office, or we may think to an Hot-Spot located in a populated conference hall, in which two couples of users send an email with a large attachment or upload their video to YouTube, as well.

Finally we point out a matter that we do not found in previous papers [3][17][68][69]: another effect of the upload aggressiveness is the dramatic increase of the packet delay in the AP. This long delay, joined with the heavy packet loss on the AP, seriously degrades not only the data downloads but also the VoIP communications, and this kind of degradation is perceived by the customers also if the duration of the upload is relatively short (e.g., 1 seconds).

Motivated by the incoming practical interest on the TCP fairness in WiFi Hot-Spot, in this Chapter we extend the literature analysis on this topic through a novel analytical model.

In addition, we propose a network framework for traffic control that is able to customize the fairness level according to the network provider need without modifying the 802.11a/b/g devices (AP and STA). We named this framework "Customizable Framework for Traffic Control (CFTC)". As a proof-of-concept of the CFTC, we provide an implementation of the CFTC that assures fairness among STAs (*STA-fairness*).

[8] We note that actually also other OSs , like Linux, use similar auto-tuning mechanisms.

We assess our models and techniques by means of an extensive testbed campaign. The tested architecture mainly consists of computers with Linux OS, USB D-Link DWL G-122 Wi-Fi cards and an Access Point Cisco Aironet 1200.

The rest of Chapter is organized as follows. In Section 2.2 we discuss the novelties introduced by this work. In Section 2.3 we describe the analytical model of the TCP fairness. In section 2.4 we depict the Customizable Framework for Traffic Control (CFTC). In Section 2.5 we present an implementation of the CFTC (in order to guarantee the targeted to the STA-fairness and prove its analytical validity. In Section 2.6 we present the testbed architecture, discuss and compare model results and testbed measures. Finally in section 2.7 we draw conclusions pointing out the still open issues.

2.2 Contribution of our work

In this section we discuss the novelties introduced in this work both in terms of fairness model and in terms of technique adopted to enforce the TCP fairness.

2.2.1 Fairness model

The fairness model here proposed addresses the well-know upstream-downstream issue and also accounts for different aspects never discussed in previous papers ([3][17][68][69]). We will describe the unfairness problem presenting several scenarios: i) TCP connections with different buffer size, ii) STAs with a different number of active TCP connections, iii) STAs with different medium access probability (i.e. different MAC implementation [72]).

Our fairness model can not be derived by the other literature models since, at the best of authors knowledge, our model is the only one that considers the possibility to have STA transmission buffer not empty. On the contrary, models such as the ones in [3], [17] and [68] assume that STA buffer is always empty. This fact cannot be neglected since the packets in the STA queue increase the round trip time of the TCP connections belonging to the STA.

We observe that this queuing aspect does not alter the validity of the previous models proposed in [3][17][68] (which consider the STA buffer always empty) since they are developed for a more restricted traffic scenario (i.e., one connection per STA and connections with the same socket buffer). Anyway, we note that whereas in these works the *empty buffer* of the STAs is a "conjecture", our model is able to proof this conjecture.

2.2.2 Technique adopted to enforce the fairness

We can find a lot of valuable approaches solving the TCP fairness issues in the literature. These approaches mainly differ in the intended goal (e.g., per-flow fairness [3][17], uplink/downlink fairness [69], etc.) and in the technique adopted to enforce the fairness (e.g., altering the TCP header [3], limiting the rate of uplink traffic [69][68], managing queuing strategies [17], etc.). What we are here proposing is a more general fairness solution, which is no more tied with a specific fairness goal but can be tailored to the provider need.

Our Customizable Framework for Traffic Control (CFTC) is an open platform designed to enforce many fairness typologies such as per-flow and per-STA fairness. Moreover, through the CFTC we can enforce several traffic control policies obtaining both fairness and service differentiations.

2.3 802.11 TCP Fairness Model

In this section we derive an analytical model of the fairness issues. In Table 4 we summarize the most of notation we adopted.

The reference scenario consists in a set of STAs which establish TCP connections with a fixed hosts. The wireless-wired bridging is performed by an AP. In addition, to make the model easily tractable, we make some simplifying assumptions reported in Table 5[9]. The discussion about the assumptions and the limits of the proposed model is reported in Appendix III.

d1	*B*: the size of AP buffer (in packets)
d2	*M*: the number of STAs
d3	Nup_i: the number of upstream connections (i.e., the data source is on the STA) of the *i*-th STA
d4	Ndw_i: the number of downstream connections (i.e., the data source is on the fixed host) of the *i*-th STA
d5	$Qdw_i(t_k)$: the number of packets stored in the *i*-th STA buffer at time t_k (i.e. at the start of round *k*) and belonging to a downstream connection
d6	$Qup_i(t_k)$: the number of packet stored in the *i*-th STA buffer at time t_k and belonging to an

[9] We will make other assumptions and definitions later in this Chapter since they can be better understood only after introducing some preliminar concepts.

	upstream connection
d7	$Q_i(t_k)$: the number of packet stored in the i-th STA buffer at time t_k, i.e. $Qdw_i(t_k)$ + $Qup_i(t_k)$;
d8	Qup_i: the average value of $Qup_i(t_k)$
d9	Qdw_i: the average value of $Qdw_i(t_k)$
d10	Q_i: the average value of $Q_i(t_k)$
d11	$Qap(t_k)$: the number of packet stored in the AP buffer at time t_k; this quantity is equal to the number of packets emitted by the AP during the k-th round
d12	$Pown\_dw_i(t_k)$: the probability that a packet transmitted on the wireless interface by the AP during the k-th *round* belongs to a downstream connection of the i-th STA.
d13	$Pown\_up_i(t_k)$: the probability that a packet transmitted on the wireless interface by the AP during the k-th *round* belongs to an upstream connection of the i-th STA.
d14	$Pown_i(t_k)$: the probability that a packet transmitted on the wireless interface by the AP during the k-th *round* is directed toward the i-th STA; i.e. $Pown_i(t_k)$= $Pown\_up_i(t_k)$ + $Pown\_dw_i(t_k)$
d15	$Pown\_up_i$: the steady-state probability that a packet transmitted by the AP on the wireless interface belongs to an upstream connection of the i-th STA. $Pown\_up_i$ is also equal to the probability that a packet stored in the AP buffer belongs to an upstream connection of the i-th STA
d16	$Pown\_dw_i$: the steady-state probability that a packet transmitted by the AP on the wireless interface belongs to a downstream connection of the i-th STA. $Pown\_dw_i$ is also equal to the probability that a packet stored in the AP buffer belongs to a downstream connection of the i-th STA
d17	$Pown_i$: the steady-state probability that a packet transmitted by the AP on the wireless interface belongs to a connectionsof the i-th STA, i.e. $Pown_i$= $Pown\_up_i$ + $Pown\_dw_i$
d18	W_i: the maximum TCP congestion window of the of the i-th STA (in packets)
d19	$Avg\_cwnd(p,W,2)$: the average value of the TCP congestion window (in packets) in presence of a segment loss probability p and in case of receiver window equals to W. This value is derived by dividing Eq. (12) of [18] by RTT, assuming T_0=RTT and b=2.
d20	$Tn(p,W,2)$: the average value of the useful segments received in an RTT in presence of a segment loss probability p and in case of a maximum congestion window equal to W. This value is derived by dividing Eq. (37) of [18] by RTT, assuming T_0=RTT and b=2.
d21	β_i: the ratio between the medium access probability of the i-th STA and the medium access probability of the AP. In case of per-packet fairness at MAC level, it results $\beta=1$; nevertheless different implementations of the 802.11 layer may alter the per-packet fairness and a STA may take a little advantage in medium access respect to the AP ($\beta>1$) or viceversa ($\beta<1$) [72].

Table 4 - Model definitions

a1	the latency and the packet loss of the wired part are neglected
a2	the STA uplink buffer is large enough as to avoid packet loss
a3	the TCP version is Reno with delayed ACK
a4	the packet loss probability at the AP downlink buffer is small enough as that: i) packet losses do not prevent the startup of TCP connections; ii) the impact of ACKs loss on the congestion window dynamic is negligible

a5	if the sum of the maximum congestion windows of all active downstream connections and of the half of the maximum congestion windows of all active upstream connections, measured in packets, is greater than the AP downlink buffer, then the AP buffer is assumed as always full
a6	the MAC layer assures a fair per-packet sharing among backlogged wireless interfaces

Table 5 - Model assumptions

The merit figures used to measure the fairness level is the *utilization factor* η_i, which is defined as the ratio between the goodput (i.e., average useful data rate on top of TCP) perceived by the *i*-th STA ($GPsta_\text{i}$) and the overall goodput transferred on the wireless interface:

$$\eta_i = \frac{GPsta_i}{\sum_{j=1}^{M} GPsta_j} \tag{1}$$

The goodput of the *i*-th STA measured in segments per second can be written as:

$$GPsta_i = \frac{Nup_i \cdot W_i}{E[D_i] + E[D_{ap}]} + \frac{Ndw_i \cdot Tn(p, W_i, 2)}{E[D_i] + E[D_{ap}]} \tag{2}$$

The first addend of Eq. (2) is the part of goodput derived by the Nup_i upstream connections. These connections fully open their congestion windows (up to the receiver window size W_i) since they do not experience segment loss (see assumption a2).

The second addend of Eq. (2) is the goodput derived by the Ndw_i downstream connections (see definition d20) which experience a packet loss probability p on the AP buffer.

The denominator of both addends in Eq. (2) is the Round Trip Time (RTT) experienced by the connections of the *i*-th STA, i.e. the sum of the average queuing delay in the AP (E[D_{ap}]) and in the *i*-th STA buffer (E[D_i]).

Let us define *normalized goodput* of the *i*-th STA ($\hat{G}Psta_i$) as the goodput of the *i*-th STA time-normalized to the average queuing delay of the AP (E[D_{ap}]), in formula:

$$\hat{G}Psta_i = GPsta_i \cdot E[D_{ap}] = \frac{Nup_i \cdot W_i + Ndw_i \cdot Tn(p, W_i, 2)}{1 + \frac{E[D_i]}{E[D_{ap}]}} \tag{3}$$

Given the assumption a6, the delay ratio E[Di]/E[DAP] is equal to the queue size ratio Qi/(βiQap), where *βi* accounts for the fact that the stations and the AP MAC layer implementations may lead to a different medium access probability [72]. As a consequence, Eq. (3) can be rewritten as:

$$\hat{G}Psta_i = \frac{Nup_i \cdot W_i + Ndw_i \cdot Tn(p, W_i, 2)}{1 + \frac{Q_i}{\beta_i \cdot Q_{ap}}} \tag{4}$$

In terms of normalized goodput, the utilization factor defined in Eq. (1) can be expressed as:

$$\eta_i = \frac{\hat{G}Psta_i}{\sum_{j=1}^{M} \hat{G}Psta_j} \tag{5}$$

In order to obtain the utilization factor η_i, in the following sections we will strive to derive the normalize goodput in Eq. (4) evaluating Q_i, Q_{ap} and p.

It is worth noting that the normalized goodput is independent of MAC layer data rate. For this reason, the analytical model of the normalized goodput that we will develop is valid for any configuration of data rate.

Moreover, although the value of the normalized goodput may be a measure not "tangible" (as it would be a bit-rate), we point out that the use of normalized goodputs leads to a tangible measure if we consider the ratio of normalized goodputs. In fact, the normalization time unit E[D_{ap}] disappears in the ratio operation and the ratio of normalized goodputs coincides with the ratio of absolute goodputs[10]. For this reason, the analytical model of the normalized goodput may result useful for several fairness studies as they are often based on goodput ratios.

2.3.1 Derivation of p, Q_{ap} and Q_i

In this section we model the evolution of the TCP connections in order to derive the AP packet loss probability (p), the average occupancy of both the AP buffer (Q_{ap}) and the STA buffer (Q_i).

We first perform the analysis under the assumption of a *lossless* AP buffer, then we repeat the analysis by considering a *lossy* AP buffer. The two formulations are

[10] For instance, this exactly occurs in Eq. (1) and Eq. (4).

combined according to the following rule: if the value of *Qap* according to the lossless formula is less than the AP buffer size, i.e. *B*, then we adopt the lossless formula; otherwise, we adopt the lossy formula.

All the next calculations are based on the definition of "round": *a round is the time interval needed to send out all the packets buffered at the AP wireless interface from the beginning of the round itself*. The *i*-th round starts at time t_{i-1} and the first round starts at time t_0. The time t_0 is any time instant after the which the system can be considered as in the steady state.

As an exception, if at the end of the *i*-th round no packet is present in the buffer of the AP wireless interface, then the (*i*+1)-th round is defined as a *void-round*. During a void-round all the backlogged STAs transmit a single packet towards the AP: after this, the void round ends.

2.3.1.1 Lossless AP buffer

In this case TCP connections fully open their congestion windows.

Let us discuss now what happens in the *i*-th STA buffer during a generic round *k* (i.e., the round that starts at time t_k).

Due to the presence of delayed-ACK mechanism [73], it is necessary to distinguish between downstream and upstream TCP connections.

In case of downstream TCP connections, when a TCP sink of a STA receives two segments from the fixed host through the AP, the STA generates one ACK directed to the fixed host and queues it in the STA buffer. As a consequence, during the *k*-th round the number of packets loaded in the STA buffer is equal to the half of number of packets received from the AP.

In case of upstream TCP connections, when a TCP source of a STA receives an ACK from the fixed host through the AP, the STA generates two segments directed to the fixed host and queues them in the STA buffer. As a consequence, during the *k*-th round the number of packets loaded in the STA buffer is equal to the double of number of packets received from the AP.

The number of packets that can leave the STA buffer during the *k*-th round is equal to the number of packets emitted by the AP in that round multiplied by $\beta_{I,}$ since a STA is able to transmit, on average, β_i packets for each packet transmitted by the AP.

If we assume a fluid flow behavior for the packet emissions and transmissions during a round, we can write the occupancy of the STA buffer at the end of round *k* (i.e., at the start of round *k*+1) as:

$$\begin{aligned} Q_i(t_{k+1}) = \max\{0, Q_i(t_k) + 0.5 \cdot Pown\_dw_i(t_k) \cdot Qap(t_k) + \\ + 2 \cdot Pown\_up_i(t_k) \cdot Qap(t_k) - \chi up_i(t_k) - \chi dw_i(t_k)\} \end{aligned} \quad (6)$$

The value $Pown\_dw_i(t_k)$ $Qap(t_k)$ is the number of packets belonging to downstream connections transmitted by the AP to the *i*-th STA. The value $Pown\_up_i(t_k)$ $Qap(t_k)$ is the number of packets belonging to upstream connections transmitted by the AP to the *i*-th STA. The value $\chi up_i(t_k)$ is the number of upstream packets leaving the *i*-th STA buffer during the *k*-th round. $\chi dw_i(t_k)$ represents the counterpart of $\chi up_i(t_k)$ in case of downstream packets. The *max* operator in (6) accounts for the obvious fact that the buffer occupancy can not be less than zero.

By assuming the random processes involved in (6) as stationery, we can derive the following approximation[11] of the average occupancy of the STA buffer from (6):

$$Q_i \approx \max\{0, Q_i + 0.5 \cdot Pown\_dw_i \cdot Qap + 2 \cdot Pown\_up_i \cdot Qap - \chi up_i - \chi dw_i\} \quad (7)$$

From Eq. (7) we argue that when the average occupancy of the STA buffer Q_i is greater than zero, then the inequality (8) holds and vice-versa. We will refer to this inequality with the term “backlogged condition”.

$$Pown_i \geq \frac{\beta_i}{2} + \frac{1}{2 \cdot Qap} + \frac{3}{4} Pown\_dw_i \quad (8)$$

Now, it is useful to separate the contribution of the upstream (Qup_i) and downstream (Qdw_i) packets to the overall average buffer occupancy Q_i. From Eq. (7) we obtain the following relationships[12] :

$$\begin{aligned} Qup_i &= \max\{0, Qup_i + 2 \cdot Pown\_up_i \cdot Qap - \chi up_i\} \\ Qdw_i &= \max\{0, Qdw_i + 0.5 \cdot Pown\_dw_i \cdot Qap - \chi dw_i\} \end{aligned} \quad (9)$$

If the STA buffer is greater than zero, the parameter χup_i can be regarded as the average number of packets leaving the STA buffer during a *round* (i.e., $\beta_i Qap$) multiplied by the probability that such packets belong to upstream connections of

[11] The approximation consists in the fact that we have considered the expected value of a max operation, i.e. E[max(X,Y)] as the max operation on the expected values, i.e. max(E[X], E[Y]). This involves an underestimation of Q_i that becomes as more negligible as more Q_i increases.

[12] We observe that, if Q_i is greater than zero and both upstream and downstream connections are present, it necessarily follows that both Qup_i and Qdw_i are greater than zero, since both the upstream and downstream packets wait in the STA queue. For this reason, the equation in (7) can be decoupled in the two equations of (9).

the *i*–th STA. This latter probability is the ratio between the average value of upstream packets in the STA buffer and the average value of all packets in the STA buffer. Repeating an analogous reasoning for the parameter χdw_i, it results[13]:

$$
\begin{aligned}
\chi up_i &= \frac{Qup_i}{Qdw_i + Qup_i} \cdot \beta_i \cdot Qap \\
\chi dw_i &= \frac{Qdw_i}{Qdw_i + Qup_i} \cdot \beta_i \cdot Qap
\end{aligned}
\tag{10}
$$

The parameter $Pown\_dw_i$ can be expressed as the ratio between the average number of packets belonging to the downstream connections of the STA *i*-th within the AP buffer and the average total number of packet contained in the AP buffer (*Qap*). Since the system is lossless, the connections fully open their congestion windows up to the receiver window size W_i. On average, there are Qdw_i TCP ACKs within the STA buffer. Consequently, the average number of packets (segments) that are in the AP buffer and belong to the downstream connections of the STA *i*-th is $Ndw_i\ W_i - 2\ Qdw_i$[14], where the factor 2 accounts for the delayed ACK. Following the same reasoning for the upstream direction, we can deduce that the average number of packets (ACKs) that are in the AP buffer and belong to the upstream connections of the STA *i*-th is $(Nup_i\ W_i - Qup_i)/2$. Thus, it results:

$$
\begin{aligned}
Pown\_dw_i &= \frac{(Ndw_i \cdot W_i - 2 \cdot Qdw_i)}{Qap} \\
Pown\_up_i &= \frac{(Nup_i \cdot W_i - Qup_i)/2}{Qap}
\end{aligned}
\tag{11}
$$

If the STA buffer is greater than zero, combining Eqs. (9), (10) and (11),leads to:

$$
\begin{aligned}
\chi up_i &= \frac{Nup_i}{Nup_i + 0.5 \cdot Ndw_i} \cdot \beta_i \cdot Qap \\
\chi dw_i &= \frac{Ndw_i}{2 \cdot Nup_i + Ndw_i} \cdot \beta_i \cdot Qap
\end{aligned}
\tag{12}
$$

[13] To take into account *void-rounds*, the value of β_i Qap should be replaced with max{β_i Qap,1}, where the max operator account for the face that during a void-round (i.e. *Qap*=0) at least one segment is sent out. So in Eq.(10) we are assuming the absence of void-rounds. This assumption may leads to an absurd solution of the system Ω (described in the following) for which *Qap*=0. In this case, in order to obtain a valid solution, we may approximate max{β_i Qap,1} with β_i Qap+1.

[14] If the RTT of the fixed network is not negligible, then ACKs are also contained in the fixed network's pipe; thus, here we must additionally subtract the number of two times the number of ACKs contained in the fixed network's pipe. The same approach must be followed for upstream connections. From this point onward, the formulation can follow the same steps also in the lossy case.

Substituting the values given by Eqs. (10)(12) in Eq. (9), the unknowns of Eq. (9) are only the parameters *Qup_i*, *Qdw_i* and *Qap*.

The occupancy of the AP buffer at the start of round *k* and its average value can be written as:

$$Q_{AP}(t_k)=\sum_{i=1}^{M}\left(\frac{Nup_i\cdot W_i-Qup_i(t_k)}{2}\right)+\left(Ndw_i\cdot W_i-2\cdot Qdw_i(t_k)\right) \tag{13}$$

$$Q_{AP}=\sum_{i=1}^{M}\left(\frac{Nup_i\cdot W_i-Qup_i}{2}\right)+\left(Ndw_i\cdot W_i-2\cdot Qdw_i\right) \tag{14}$$

In order to derive the average value of the occupancy ($Q_i=Qup_i+Qdw_i$) of the buffer of the *i*-th STA and of the AP (*Qap*), we can consider the equation system composed by Eqs (14) and (9) evaluated for each *i*. In doing so, we obtain an equation system (named Ω) of 2*M*+1 equations in 2*M*+1 unkowns (Qup_i, Qdw_i and *Qap*).

Unfortunately, the *max* operator within the (9) makes the system Ω not-linear and difficult to resove. To face this issue, the system Ω may be resolved through a numerical computation [74] searching the solution by varying *Qap* in the range $0\le Qap\le\sum_{i=1}^{M}\left((Nup_i\cdot W_i)/2+Ndw_i\cdot W_i\right)$, where the upper bound represents the number of packet within the AP buffer when all STA buffers are empty. As an alternative to this numerical computation, we propose another approach for solving the system Ω in Appendix IV.

As case study, we report the analytical solution of the system in case of $\beta_i=\beta=1$ i.e. when the MAC layer implementations do not introduce unfairness in the medium access. In this case, based on Eq. (8), we can state that, if the *i*-th STA is backlogged, then $Pown_i \ge 0.5$. This implies that only one STA might be backlogged[15]. Assuming that the backlogged STA is the *i*-th STA and using $Q_j=0$ for all other STAs, according to Eq. (8) the *i*-th STA is backlogged if and only if:

$$Nup_i\cdot W_i\ge\gamma+Ndw_i\cdot W_i \tag{15}$$

where

[15] This strictly occurs even if $\beta\ge 1$

$$\gamma = \left(\sum_{\substack{j=1 \\ j \neq i}}^{M} Nup_j \cdot W_j + 2 \cdot Ndw_j \cdot W_j \right) \tag{16}$$

If the *i*-th STA verifies the backlogging condition in Eq. (15), then the average STA buffer occupancy of the *i*-th STA can be formulated as :

$$Q_i = \max\left\{ 0, \frac{(Nup_i + 0.5 \cdot Ndw_i) \cdot (Nup_i \cdot W_i - Ndw_i \cdot W - \gamma)}{Nup_i - Ndw_i} \right\} \tag{17}$$

2.3.1.2 Lossy AP buffer

This case differs from the previous one in the following aspects:

i) as regards downstream connections, the loss of a segment in the AP buffer reduces the TCP congestion window. Hence, the congestion window of a downstream connection is no more constant, as in the *lossless* case, but it depends on the packet loss probability of the AP buffer;

ii) as regards upstream connections, the loss of ACKs in the AP buffer implies that, when a TCP source located in a STA receives an ACK after a sequence of ACK losses, the TCP source sends out a burst of TCP segments (ACKs being cumulative). The size of this burst is equal to the number of segments cumulatively acknowledged by the received ACK. It follows that the STA may queue more than one packet for each received ACK in its buffer;

iii) from assumption a5, the value of *Qap* is no more an unknown but it is equal to *B*.

Let us first consider the impact of the downstream connections on the STA buffer. Like in the lossless case, every time two segments are received by a TCP sink of a STA, the TCP sink sends out the relevant ACK[16]. This means that Eq. (9) holds for the downstream packets also in the lossy case.

Let us now consider the upstream connections of a generic STA. We remind that, under the assumption a4, the congestion window of upstream connections is equal to the receiver window W_i. Thus, the overall number of in-fly segments is equal to $Nup_i\ W_i$. Given the assumption a1, these packets can be either in the AP buffer or in

[16] This is strictly true in absence of losses. In fact, when a segment loss occurs during the fast recovery phase, a duplicated ACK is sent for each segment. We do not consider this behavior in our model and we do not asses its impact in the testbed.

the STA buffer or they are lost at the AP buffer. Packets within the STA buffer are TCP segments, whereas lost packets or packets in the AP buffer are TCP ACKs. This implies that, at the start of round *k*, the number of in-fly ACKs belonging to the *i*-th STA is equal to ($Nup_i\ W_i$ - $Qup_i(t_k)$)/2.

Since ACKs are cumulative, at the end of round *k*, $Nup_i\ W_i$ - $Qup_i(t_k)$ segments will be acked and the TCP senders will queue the same number of segments in the STA buffer.

As a consequence, equation (9) in the lossy case may be formulated as:

$$\begin{aligned} Qup_i &= \max\{0, Qup_i + (Nup_i \cdot W_i - Qup_i) - \chi up_i\} = \max\{0, Nup_i \cdot W_i - \chi up_i\} \\ Qdw_i &= \max\{0, Qdw_i + 0.5 \cdot Pown\_dw_i \cdot B - \chi dw_i\} \end{aligned} \tag{18}$$

Due to the presence of segment loss, the value of the probability $Pown\_dw_i$ differs from the value obtained in lossless case (see Eq. (11)). This probability can be computed as the ratio between the number of downstream packets of the *i*-th STA entering the AP buffer in the unit time and the overall number of packets entering the AP buffer in the unit time. If we consider the average packet delay in the AP buffer $E[D_{AP}]$ as time unit (i.e., the average duration of a round), the following equation holds:

$$Pown\_dw_i = \frac{\dfrac{Ndw_i \cdot Avg\_cwnd(p,W_i,2)}{E[D_{AP}]+E[D_i]} \cdot E[D_{AP}]}{B \cdot \left(\dfrac{1}{1-p}\right)} = \frac{Ndw_i \cdot Avg\_cwnd(p,W_i,2) \cdot (1\text{-}p)}{B + \left(\dfrac{Q_i}{\beta_i}\right)} \tag{19}$$

where $E[D_i]$ is the average packet delay of the buffer of the *i*-th STA, *p* is the steady-state packet loss probability at the AP buffer and Avg_cwnd(p,W) is the average value of the TCP congestion window when there is a loss probability *p*.

Let us consider now the case in which the average buffer occupancy of the STA is greater than zero. If we solvin the downstream equation reported in (18) and use Eq. (10), we obtain:

$$0.5 \cdot Pown\_dw_i \cdot B = \chi dw_i = \frac{Qdw_i}{Qdw_i + Qup_i} \cdot B \cdot \beta_i \tag{20}$$

Likewise, if we solve the upstream equation in (18) and use Eq. (10) we obtain:

$$Qup_i = Nup_i \cdot W_i - \frac{Qup_i}{Qdw_i + Qup_i} \cdot B \cdot \beta_i \tag{21}$$

By combining Eqs. (20) and (21), the average value of the STA buffer can be expressed as:

$$Q_i = Qdw_i + Qup_i = \beta_i \cdot \frac{Nup_i \cdot W_i}{\left(\beta_i - Pown\_dw_i/2\right)} - B \cdot \beta_i \qquad (22)$$

If we use the value of *Pown_dw$_i$* as evaluated by Eq. (19) in Eq. (22), we obtain a quadratic equation in the unknown Q_i that admits only a valid solution: $Nup_i\ W_i + 0.5 \cdot Avg\_cwnd(p, W_i, 2)(1-p) - \beta_i B$. As then Eq. (22) has been derived under the condition $Q_i > 0$, the average value of the buffer of the i-th STA can be formulated as:

$$Q_i = \max\left\{0, Nup_i \cdot W_i + \frac{1}{2} \cdot Ndw_i \cdot Avg\_cwnd\left(p, W_i, 2\right) \cdot \left(1-p\right) - \beta_i \cdot B\right\} \qquad (23)$$

To complete the evaluation of Q_i, we need to evaluate the packet loss probability p. By definition, p is equal to one minus the ratio between the traffic leaving the AP buffer and the traffic offered to the AP buffer:

$$p = 1 - \frac{\dfrac{B}{E[D_{AP}]}}{\displaystyle\sum_{j=1}^{M} \frac{0.5 \cdot \left(Nup_j \cdot W_j\right) + Ndw_j \cdot Avg\left(p, W_j, 2\right)}{E\left[D_j\right] + E\left[D_{AP}\right]}} = 1 - \frac{B}{\displaystyle\sum_{j=1}^{M} \frac{0.5 \cdot \left(Nup_j \cdot W_j\right) + Ndw_j \cdot Avg\left(p, W_j, 2\right)}{1 + \dfrac{Q_j}{\beta_j \cdot B}}} \qquad (24)$$

If we combine Eq. (24) with the *M* Eqs. (23) (one for each of *i* with $1 \le i \le M$), we obtain a non-linear system of a *M*+1 equations with *M*+1 unknowns (Q_i and *p*); we refer to this system by the symbol Ψ.

We can resolve the system Ψ through a numerical computation [74], which searches the solution by varying *p* in the range $0 < p < 1$.

Finally, Eq. (25) reports a practical approximation for the average buffer occupancy of a STA in the lossy case, where the impact of the downstream connections in (23) is neglected. The approximation can be explained as follows: when queuing phenomena occur in the STA, the AP buffer is heavy loaded and the downstream connections work with very small congestion window.

$$Q_i = \max\left\{0, Nup_i \cdot W_i - \beta_i \cdot B\right\} \qquad (25)$$

2.3.1.3 Qualitative insights on the average occupancy of the STA buffer

In this section we derive "qualitative" insights about the average occupancy of the STA buffer. These insights will be useful to understand the results obtained in the testbed experiments. They are basically derived from Eq. (8) and (23). To make this qualitative analysis more tractable, we assume $0.5<\beta_i<2$. In our opinion, the considered range for β_i is reasonable since it reflects the most of real life cases.

As first insight, we state that: *if a STA has not upstream connections, then the STA buffer is practically empty .*

As second insight, we state that: *if a STA has upstream connections, then the STA buffer is not-empty when the number of segments introduced by the STA in the network overcomes a certain threshold (named "backlogging" threshold).*

The backlogging threshold is different depending on the lossless and lossy case. In the lossless case the backlogging threshold linearly depends on the number of segments relative to the connections of the other STAs (e.g., see γ in Eq. (15)); in the lossy case the backlogging threshold linearly depends on the size of the AP buffer (see Eq. (25)).

Thus, queuing phenomena in the STA buffer are essentially produced by upstream connections. We can qualitatively explain this point by observing that the transmission of two downstream packets (i.e., TCP segments) from the AP implies the queuing of a TCP ACK in the STA buffer and grants at the same time two transmission opportunities to the STA. These opportunities allow the STA to send all the queued ACKs. On the contrary, when the AP transmits an upstream packet (i.e., a TCP ACK), the STA gets a transmission opportunity but, since the ACK may confirm more than one packet, the STA queue may be filled with a number of packets greater than the transmission opportunities granted to the STA.

2.4 Custom Framework for Traffic Control (CFTC)

In order to allow service providers to personalize the level of service offered in a Wi-Fi hot-spot, we designed a network framework, named Customizable Framework for Traffic Control (CFTC), which enforces customizable traffic control policies in 802.11 WLAN operating in DCF mode.

CFTC can be deployed without any modification of the 802.11 MAC layer, wireless STAs and APs.

Since modifying both TCP and 802.11 to solve the fairness problem is not practical, we propose to delegate the enforcement of the traffic control to a “third actor”.

As depicted in Figure 31, a Virtual BottleNeck (VBN) plays such a role of “third actor” in the proposed framework. More precisely, the VBN is logically placed between the AP and the Internet Gateway, or within one of them.

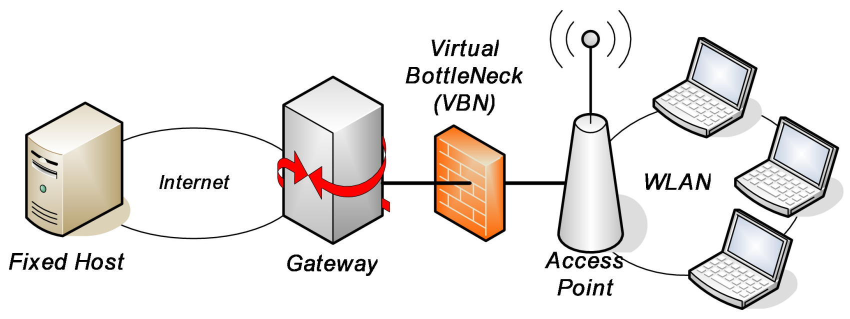

Figure 31: Network sketch of the Customizable Framework for Traffic Control (CFTC)

The underling concept is shown in Figure 32. The upper scheme of Figure 32 represents the path followed by the TCP flows from the Internet Gateway to the STA in absence of VBN. A downstream TCP flow passes through a pipe that is the wired part, and then it passes through another pipe that is the wireless part. The width of a pipe is related to the capacity of the related network medium. The wireless pipe is the path bottleneck. In this bottleneck, the 802.11 regulates the traffic as described by the previous Section 2.3.

The lower scheme of Figure 32 represents what happens in presence of the VBN. The VBN limits the transfer rate of the wired part to a capacity C-ε (bps), that is a bit lower than the wireless capacity C (bps). In addition, as it occurs in the wireless interface, the VBN is devised so that *the capacity C-ε is shared among uplink and downlink packets*. In this way the wireless pipe does not represent the path bottleneck anymore. The wireless interface can now be considered unloaded and the 802.11 should become a “transparent” pipe, since the TCP streams adapt their rates to the capacity available in the network bottleneck. As a consequence, the resource contention only occurs in the VBN which is a centralized point and may be easily managed and customized by the service provider.

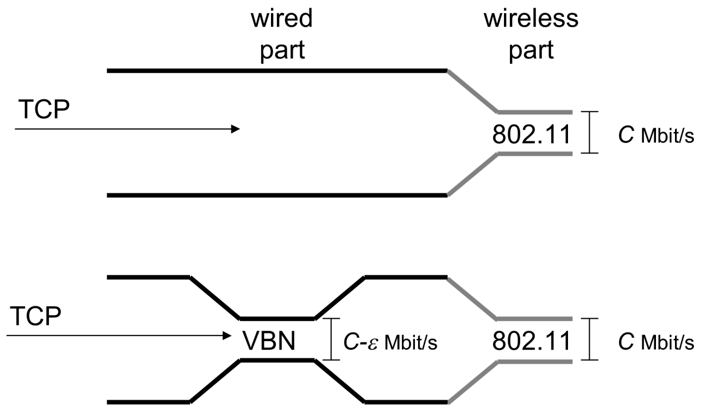

Figure 32: Conceptual sketch of the Customizable Framework for Traffic Control (CFTC)

The generic architecture of the VBN is drawn in Figure 33 in which we can identify a classifier, a scheduler and a WLAN Capacity Estimator (WCE).

The WCE derives the capacity C-ε from an estimation algorithm. The value C-ε should be as greater as possible to limit wastes of radio capacity, but it must also assure a very small number of packets in the AP queue.

The classifier classifies the incoming packets in a set of queues. Each queue is dedicates to a specific class of traffic. Hence, the number of queues depends on the traffic differentiation policy adopted by the service provider.

Finally, the scheduler drains packets from queues with a rate of C-ε and the queue selection is done according to the traffic policy adopted by the service provider.

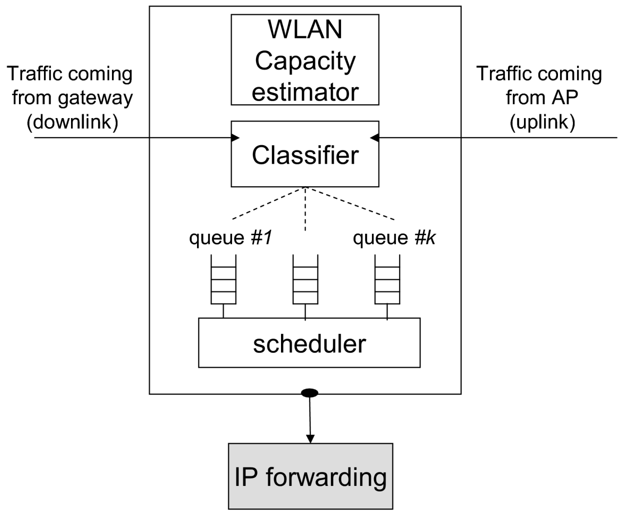

Figure 33: Generic functional architecture of the Virtual BottleNeck (VBN)

Although the CFTC is originally developed for TCP traffic, it might also be used to control real-time traffic (by introducing specific queue for UDP flows).

2.5 Achieving STA Fairness through CFTC

In this Section we show how to customize the framework CFTC in order to obtain "STA-fairness". With regard to a Wi-Fi Hot Spot with *M* STAs, i.e. *M* users, STA-fairness means that all the users obtain the same transfer capacity. In other words, if we consider the network scenario of Section 2.3, guaranteeing STA-fairness means that given two stations, STA *i* and *j*, the ratio between the utilization factors η_i and η_j (see Eq. (1)) is equal to 1 for each couple *i,j*.

The VBN architecture that we propose with the goal to achieve STA-fairness is depicted in Figure 34. There is a queue for each STA and the scheduler is a Deficit Round Robin (DRR) on a byte basis. The queues are large enough as to avoid packet loss. The choice of large STA queues is due to the fact that small queues would involve segment losses and reduce the achievable useful data rate.

DRR makes the system work-conserving; i.e., resource wasting is avoided.

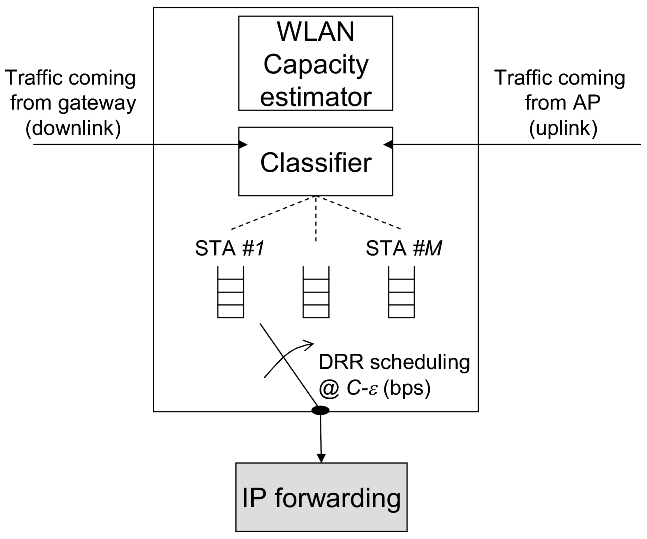
WLAN Capacity estimator
Traffic coming from gateway (downlink)
Traffic coming from AP (uplink)
Classifier
STA #1
STA #M
DRR scheduling @ C-ε (bps)
IP forwarding

Figure 34: Architecture of the VBN for STA-fairness

In Appendix V we derive a simple analytical model which proves the effectiveness of the VBN architecture reported in Figure 34. We obtain that the overall goodput perceived by the *i*-th STA is equal to:

$$GPsta_i = \frac{(Nup_i \cdot W_i + Ndw_i \cdot W_i) \cdot Lpayload}{RTT_i} = \frac{Lpayload}{Lseg} \cdot \frac{2 \cdot Lseg}{2 \cdot Lseg + Lack} \cdot \frac{C - \varepsilon}{M} \tag{26}$$

This value is independent of the index *i* and, consequently, all the STAs experience the same goodput.

We verified the effectiveness of the VBN architecture reported in Figure 34 by assuming that WCE is able to exactly determine the capacity C-ε. In our opinion, the estimation algorithm is a topic which requires an extensive analysis and we prefer to not cope also with this issue at the moment.

We conclude the section by highlighting that the proposed structure of the VBN can be easily modified (for what concerns the queues) to cope with other fairness policies. For example, in order to achieve flow-fairness (as done in [3][17]), a queue for each flow has to be implemented in the VBN. In order to achieve Upstream/Downstream fairness, as done in Chapter 1, we should instead use two VBN queues: one for upstream and the other one for downstream traffic.

2.6 Testbed

In this section we assess the effectiveness of i) the TCP model presented in Section 2.3 and ii) the VBN proposed in 2.5. First of all, we describe the testbed setup, then we present the numerical results.

2.6.1 Testbed setup

The architecture of the testbed is sketched in Figure 35. To make the description more readable, in this section we refer to the testbed architecture as formed by three Wi-Fi cards; although we used six Wi-Fi cards in the measurement campaign.

Differently from our previous work [75] where each Wi-Fi card was hosted by a different notebook, in this testbed we used a set of USB Wi-Fi cards, all attached to a unique powerful Linux PC (kernel 2.6.17) named "STA PC". This configuration realizes a compact testbed completely controlled by the STA PC through "*ssh*" communications.

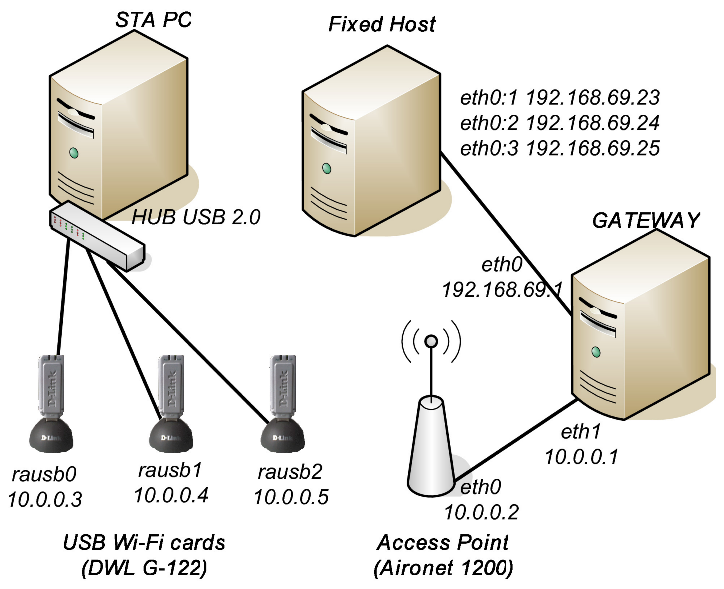

Figure 35: Network architecture of the testbed

The Wi-Fi cards are DLink DWL G-122 with Ralink chipset, firmware version 3.0 and hardware version C1. We used the open Linux driver RT73 v1.0.3.6 downloadable by the Ralink web-site. We modified the driver in order to select manually the physical transmission rate used in the communications with the AP [17]. Such a modification allows the realization of multi-rate scenarios in the testbed.

The Access Point is a Cisco Aironet 1200 configured to operate in 802.11b mode. We derived the size of the AP buffer (toward the wireless interfaces), that is 75 packets, from its technical specifications.

The AP is connected to a Linux Desktop PC (named “GATEWAY”) through an Ethernet cable. The GATEWAY applies the rules of the Internet Gateway and it also implements the Virtual BottleNeck software. On the other side, the GATEWAY is connected to another Linux Desktop PC that acts as the Fixed Host.

For what concern the addressing scheme, the STA PC belongs to the network 10.0.0.0/24 and the Fixed Host belongs to the network 192.168.69.0/24. The internetworking between the two addressing spaces is performed by the GATEWAY.

The IP addresses of the network interfaces are reported in Figure 35, where `rausb%x` is the logical name of the `x`-th Wi-Fi interface of the STA PC and

[17] The novel *iwpriv* command that we implemented changes the “capability set” communicated by the Wi-Fi card to the AP during the association phase. In doing so, we control the rate at which the AP transmits to the Wi-Fi card. For the reverse direction, the Linux shell provides by itself the command `iwconfig interface rate`. Finally, we note that we have rectified the driver from two bugs: the first one prevented the use of more than two Wi-Fi cards on the same PC; the second one led the card to use CWmin=15 even in case of 11b physical layer, instead of CWmin=31.

`eth%x` is the logical name of the `x`-th Ethernet interface. In addition, we use multiple IP addresses on the `eth0` Ethernet interface of the Fixed Host.

DEVICE	DESTINATION	GATEWAY	NETMASK	INTERFACE
STA PC	192.168.69.23	10.0.0.1	255.255.255.255	rausb0
STA PC	192.168.69.24	10.0.0.1	255.255.255.255	rausb1
STA PC	192.168.69.25	10.0.0.1	255.255.255.255	rausb2
Fixed Host	10.0.0.3	192.168.69.1	255.255.255.255	eth0:1
Fixed Host	10.0.0.4	192.168.69.1	255.255.255.255	eth0:2
Fixed Host	10.0.0.5	192.168.69.1	255.255.255.255	eth0:3
GATEWAY	10.0.0.0	10.0.0.1	255.255.255.0	eth1
GATEWAY	192.168.69.0	192.168.69.1	255.255.255.0	eth2

Table 6 - Routing Table of testbed devices

The routing table used in the testbed devices is reported in Table 6.

The ARP table of the GATEWAY has statically set.

To perform the measurements, we use the *iperf* tool [18]. For instance, to reproduce a traffic scenario where each of three different STAs has one downstream TCP connection, we run three different iperf client instances on the Fixed Host with destination addresses 10.0.0.3, 10.0.0.4, 10.0.0.5, and we run three different iperf server instances, bound with the interfaces 10.0.0.3, 10.0.0.4 and 10.0.0.5, on the STA PC.

In each test TCP connections last three minutes and are randomly established in the first two seconds. The performance of a TCP connection has been gathered on the iperf server side by recording the goodput of the connection every 5 seconds on a text file. This file has been processed through a PERL script in order to make it readable for the MATLAB tool, which we used to analyze the collected measurements. We also exclude the first and last 10 seconds of the run to evaluate the average of the measurements.

We repeated the measurements of a specific test ten times and we evaluated the average performance and its 95% confidence interval; sometimes this interval is so small as to be not visible in the performance plots hereafter reported.

As far as the implementation of the VBN in Figure 34 is concerned, we realized the VBN as a user-space multi-thread C++ program that run on the GATEWAY. In order to handle packet transmission and reception at user space, we used the LIBIPQ library. The code of the Deficit Round Robin (DRR) has been gathered

[18] In addition, in some scenarios in which we measured the delay, we used the PING tool.

from the Network Simulator 2 source code. The quantum of the DRR is 100 bytes and the overall buffering capacity of the DRR is 1000 packets.

Finally, as discussed in Section 2.5, we did not devised the WLAN Capacity Estimator (see Figure 34), and we manually set the VBN by providing the parameter $C\text{-}\varepsilon$ as input.

2.6.2 Testbed results

We measured the system performance in presence and absence of the VBN in different scenarios that we describe in the following sub-sections. These scenarios mainly show what is the impact of the number of connections per STA, the TCP receiver window and the Wi-Fi transmission rate of the STAs (19) on TCP fairness. In addition, we will also show what happens to a VoIP call in presence of TCP connections.

In each scenario we assessed the validity of the analytical model described in Section 2.3, we proved the effectiveness of the CFTC in achieving STA-fairness through the use of the VBN implementation and, finally, we monitored eventual inefficiencies introduced by the VBN.

As far as the computation of the parameters β_i (see def. d21) used in the model of Section 2.3, we observe that our testbed hardware is formed by Wi-Fi cards of the same type; therefore β_i=β for each value of *i*. For the computation of the parameter β we have set up one downstream UDP connection and one upstream UDP connection on the STA n.1 (with transmission rate of 11 Mbps). Each UDP connection is loaded by a CBR source that generates packets 512 bytes long with a rate of 10 Mbps, in such a way that each UDP connection is able to saturate the overall wireless capacity. We evaluated β as the ratio between the upstream UDP goodput and the downstream UDP goodput. We repeated such a measure ten times obtaining an average value of 1.18. In the following, we consider β_i=β=1.18.

Regarding the selection of the bit-rate $C\text{-}\varepsilon$ of the VBN (Figure 34), in case of single-rate scenarios at 11 Mbps, we measured that a value of 5.8 Mbps is the practical bound over which packet loss occurs at the AP buffer. For this reason, we have assumed the value 5.8 Mbps as the WLAN Capacity *C*. Consequently, since

[19] We also analyzed the impact of the number of STA by assuming that each STA has a single TCP connection. The obtained results show that the system assures STA-fairness regardless of the presence of VBN.

we need also to limit the queuing phenomena on the AP, we have empirically selected the value $C\text{-}\varepsilon$ =5.6 Mbps. We remark that, in case of different VBN implementations that lead to a greater traffic burstiness, it might be necessary to further decrease the value of $C\text{-}\varepsilon$ in order to limit the number of packets in the AP buffer. Indeed, under a constant load, the more bursty the traffic, the more present the queuing phenomena.

In multi-rate scenarios, we adopted the same approach for the evaluation of $C\text{-}\varepsilon$. Anyway, in this cases the use of a fixed values for all the test is not reasonable since the available wireless capacity strongly depends on the specific traffic pattern. For this reason, we repeated the measuring of $C\text{-}\varepsilon$ and the selected values are reported in the relative following sub-sections.

2.6.2.1 Impact of the number of downstream TCP connections per STA

We considered five STAs with physical transmission rate set to 11 Mbps. The number of downstream TCP connections of STA n.1 ranges from 1 to 6, while the other STAs have only one downstream connection. The TCP receiver window of all the connections is equal to 42 segments (i.e., about 65 kB).

We selected the ratio between the utilization factor η_1 of the STA n.1 (see Eq. (1)) and the utilization factor η_2 of the STA n.2 as TCP fairness index. This ratio is also the ratio between the goodput of the STA n.1 and the goodput of the STA n.2. Moreover, we observed that the STA n.2 has the same traffic pattern as the STAs n.3,4,5, therefore the ratio η_1/η_2 is expected to be equal to any ratio η_1/η_j with $2 \leq j \leq 5$.

Figure 36 shows the utilization ratio η_1/η_2 obtained in absence of VBN: i) by means of measurements (labeled in the following as “measures free”); ii) by the model output (labeled in the following as “model free”). Figure 36 also shows the utilization ratio η_1/η_2 in presence of VBN obtained by means of the measurements (labeled in the following as “measures with VBN”).

Finally, in Figure 36 we plot the *cumulative goodput* obtained in absence of the VBN (labeled in the following as “avg. GP free”) and in presence of the VBN (labeled in the following as “avg. GP VBN”). The average cumulative goodput is computed evaluating the sum of the goodput of all the STAs for each considered case reported in the abscissa axis. The average cumulative goodput is the average value of these sums.

Note that η_1/η_2 equal to 1 implies a perfect STA-fairness.

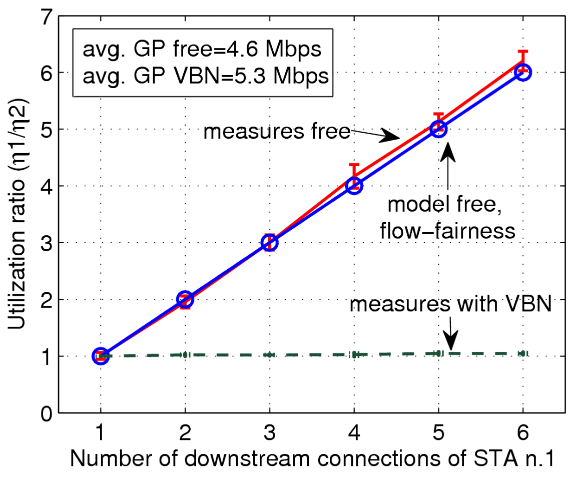

Figure 36: Utilization ratio η1/η2 versus number of downstream connections on STA n. 1, other STAs have 1 downstream connection, all STAs with physical transmission rate at 11 Mbps and maximum TCP receiver window 42 pkts (65 kB)

Let us discuss now the results obtained in Figure 36. Without VBN, there are not queuing phenomena on the STAs buffers, since there are only downstream connections. Each TCP connection experiences the same RTT and the same packet loss probability; consequently, each connection perceives the same goodput. The system is providing flow-fairness and when the STA n.1 has a number X of connections, its goodput is X-time greater than the goodput of the STA n.2; i.e., $\eta_1/\eta_2 = X$. In presence of the VBN, we ascertain the VBN effectiveness in reporting the fairness at STA level. The effectiveness of the VBN also occurs in all the other cases which we will analyze subsequently. For this reason we avoid to further comment this fact. The average cumulative goodput with VBN (5.3 Mbps) is little greater than the one obtained without VBN (4.6 Mbps). This is due to the fact that the system is lossless with VBN. On the contrary, without VBN segments losses occur in the AP buffer, the TCP send-rate floats and the average goodput is lower. Figure 37 reports the results obtained repeating the same tests as Figure 36 but in a multi-rate environment. The STA n.1 has a physical transmission rate of 2 Mbps while the others four STAs work at 11 Mbps. As expected [76], the level of fairness does not change, since the MAC layer grants transmission opportunities independently of the transmission rate. This test assesses the effectiveness of the model proposed in Section 2.3 and of the VBN approach in case of multi-rate environment. As previously mentioned, the bit-rate of the VBN (i.e., C-ε) used in this multi-rate test is not a constant and the values adopted for the six cases in the abscissa axis are: 3.5, 3.2, 2.8, 2.8, 2.8, 2.8 Mbps.

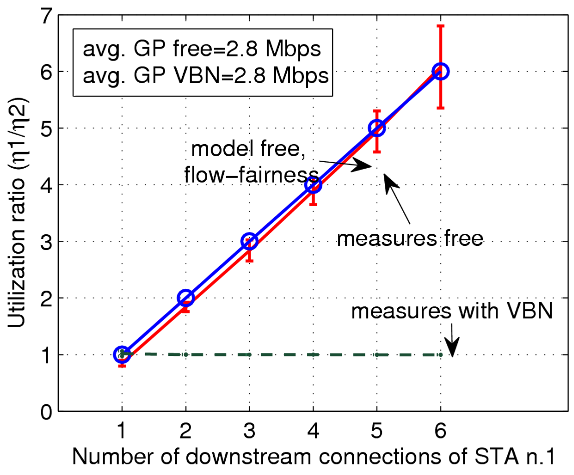

Figure 37: Utilization ratio η1/η2 versus number of downstream connections on STA n. 1, other STAs have 1 downstream connection. STA n.1 with physical transmission rate at 2 Mbps, others STAs at 11 Mbps. Maximum TCP receiver window 42 pkts (65 kB)

2.6.2.2 Impact of the number of upstream TCP connections per STA

We consider five STAs with physical transmission rate set to 11 Mbps. The number of upstream TCP connections of STA n.1 varies from 1 to 6, while the other STAs have only one upstream connection. The TCP receiver window of all connections is equal to 42 segments (i.e., about 65 kB).

The results obtained in this scenario are reported in Figure 38. Without VBN, there are not queuing phenomena on the STAs n. 2,3,4,5. On the contrary, if the number of connections is greater than 2, queuing phenomena on the buffer of the STA n.1 show up and the queue length increases with the number of connections. Therefore, with respect to the other STAs, the connections of the STA n.1 experience a greater RTT that reduces the aggressiveness of TCP connections.

This case is useful to understand the impact of queuing phenomena in the STA buffer since the system would operate in flow-fairness fashion in absence of such a phenomena.

We point out that the ratio η_1/η_2 tends to be constant in Figure 38. The asymptotic value can be derived using the Eq. (5) for η_1 and η_2 and operating the limit of η_1/η_2 (considering $N_{up1} \rightarrow \infty$). The result of the limit operation is $\beta_i B/W_i$, i.e. about 2.1.

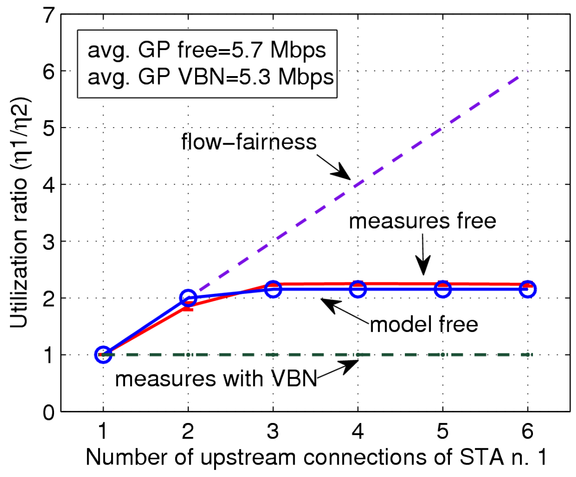

Figure 38: Utilization ratio η1/η2 versus number of upstream connections on STA n. 1, other STAs have 1 upstream connection, all STAs with physical transmission rate at 11 Mbps and maximum TCP receiver window 42 pkts (65 kB)

The average cumulative goodput with VBN (5.3 Mbps) is little lower than the one obtained without VBN (5.7 Mbps). This is due to the fact that in both cases the losses of TCP connections regard only ACKs and the TCP connections fully open their congestion windows. In addition, in presence of VBN the capacity of the communicative path is a little restricted.

Figure 39 reports the results obtained repeating the same tests as Figure 38 but in a multi-rate environment. As said before, the presence of STAs with different physical transmission rate does not change the fairness level.

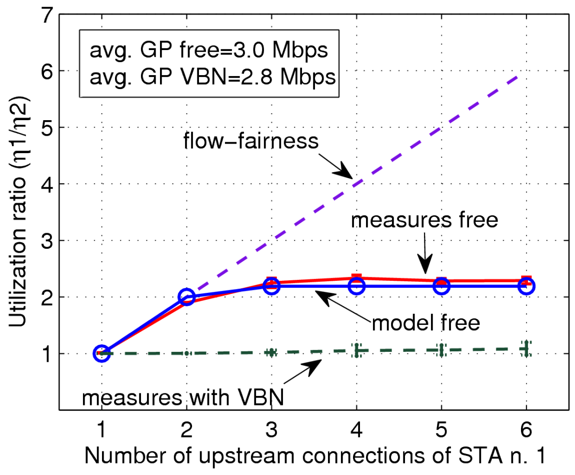

Figure 39: Utilization ratio η1/η2 versus number of upstream connections on STA n. 1, other STAs have 1 upstream connection. STA n.1 with physical transmission rate at 2 Mbps, others STAs at 11 Mbps. Maximum TCP receiver window 42 pkts (65 kB)

2.6.2.3 Impact of the downstream TCP connection receiver windows

We consider five STAs with physical transmission rate set to 11 Mbps. Each STA has one downstream connection. The TCP receiver window of the STA n.1 varies from 6 packets (i.e., 8 kB) to 42 packets (i.e., 65 kB). Regarding the TCP receiver window of the other STAs, we consider two sub-cases: 6 and 42 packets. The first sub-case analyzes the TCP fairness versus the receiver window in environments in which operative systems using a small receiver window (e.g., Microsoft Windows XP) are the majority. On the contrary, the last sub-case (i.e., 42 packets of receiving window) accounts for environments in which operative systems using large receiver window (e.g., Microsoft Windows Vista or latest Linux Kernels) are the majority.

STAs 2÷5 with TCP receiver window equals to 6 packets (8 kB)

The results obtained in this case are reported in Figure 40.

Without VBN there are not queuing phenomena on the STAs, since there are only downstream connections. Therefore, each TCP connection experiences the same RTT.

Moreover, due to the small size of the TCP receiver window (i.e., 8 kB) of the STAs 2,3,4,5, the number of in-fly packets is lower than the size of the buffer of the AP; thus the AP does not loss packets. Consequently, the goodput of a TCP connection (and of a STA this case) is equal to the receiver window over RTT and the utilization ratio η_1/η_2 is equal to the ratio W_1/W_2.

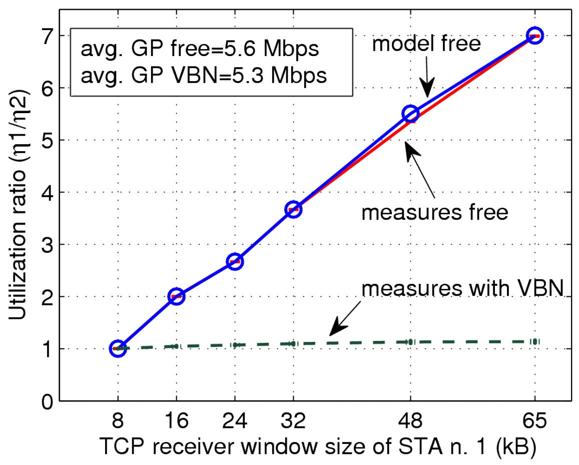

Figure 40: Utilization ratio η1/η2 versus the size of the TCP receiver window of STA n. 1, other STAs have TCP receiver window equal to 8 kB (i.e. 6 pkts). All STAs have only one downstream connection and the physical transmission rate is 11 Mbps

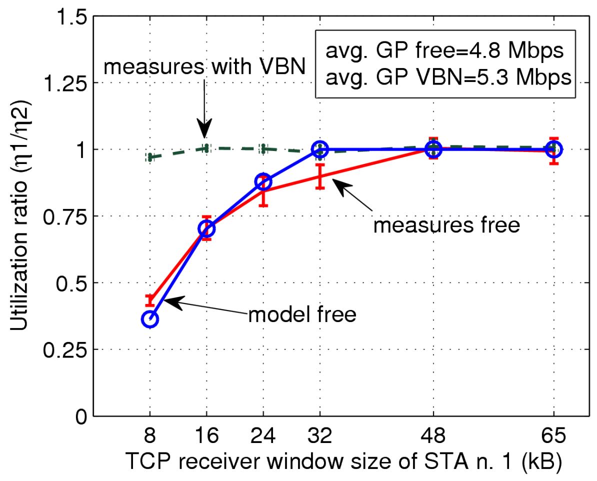

Figure 41: Utilization ratio η1/η2 versus the size of the TCP receiver window of STA n. 1, other STAs have TCP receiver window equal to 65 kB (i.e. 42 pkts). All STAs have only one downstream connection and the physical transmission rate is 11 Mbps

STAs 2÷5 with TCP receiver window equals to 42 packets (65 kB)

The results obtained in this case are reported in Figure 41.

As in the previous case, without VBN there are not queuing phenomena on the STAs since none STA overcomes the backlogging threshold (Section 2.3.1.3). Therefore, each TCP connection experiences the same RTT. Nevertheless, due to the large size of the receiver window (i.e., 65 kB) of the STAs 2,3,4,5, segments losses occur in the AP buffer.

To better understand the results of Figure 41, it is useful to discuss the impact of the receiver window size on the TCP goodput. Let us assume that *W** is the average congestion window reached by the TCP when the receiver window is unlimited. If the size of the receiver window is a value *W* enough greater than *W**, then the TCP performance are reasonably independent of the value of *W*; indeed, the congestion window is limited only by the segment losses. On the contrary, if *W* is smaller than *W** or not enough greater, the limitation of the congestion window (and of the TCP performance) is also due to the small size of the receiver window.

In Figure 41 for values of the receiver window greater than 32 kB, the congestion window (and the goodput) of TCP connections are limited only by segments losses. Hence, the TCP connections of the STA n.1 and of STA. n.2 have the same average congestion window and, since the RTT is also the same, STA n.1 and STA. 2 have also the same goodput; i.e., $\eta_1/\eta_2=1$.

On the contrary, for values of the receiver window lower than 32 kB, the small receiver window used by the STA n.1 limits the congestion window. The smaller the size of receiver window, the smaller the average congestion window, the TCP goodput and the utilization η_1 of the STA n.1. For this reason, the ratio η_1/η_2 decreases for values of the receiver window lower than 32 kB.

We observe a little deviation of the model result (label "model free") from the testbed results (label "measures free") near to the value of W_1=32 kB. This is due to the fact that the adopted formula overestimates the average congestion window $Avg\_cwnd(p,W,2)$ (see definition d19) for value of W near to W^* and, in this case, W^* is about 32 kB.

Finally, the average cumulative goodput with VBN (5.3 Mbps) is greater than the one obtained without VBN (4.8 Mbps) since without VBN the TCP connections lose segments leading to a reduction of the average goodput.

2.6.2.4 *Impact of the upstream TCP connection receiver windows*

Considering upstream TCP connections, the scenario is dual to the one presented in the previous sub-section.

When the connections are upstream, none segment loss occurs and the congestion window of each connection reaches the size of the receiver window. Anyway, queuing phenomena may occur on the STA buffers and this differentiates the value of the RTT experienced by the connections.

As previously done, we consider two sub-cases that differ in the value of the TCP receiver window of the STAs n.2,3,4,5.

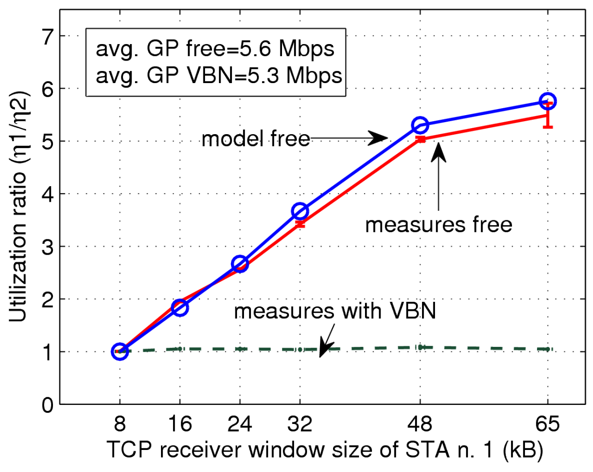

Figure 42: Utilization ratio η1/η2 versus the size of the TCP receiver window of STA n. 1, other STAs have TCP receiver window equal to 8 kB (i.e. 6 pkts). All STAs have only one upstream connection and the physical transmission rate is 11 Mbps

STAs 2÷5 with TCP receiver window equals to 6 packets (8 kB)

The results obtained in this case are reported in Figure 42.

Without VBN there are not queuing phenomena on the STA n. 2,3,4,5. Instead, for values of receiver window greater than 48 kB, the STA n.1 starts to enqueue packets in its buffer. It follows that for values of receiver window smaller than or equal to 48 kB all the connections have the same RTT and the utilization ratio η_1/η_2 is equal to the ratio of the receiver windows W_1/W_2. Increasing the receiver window, the aggressiveness of the connection of STA n.1 is reduced by the fact that it experiences a greater RTT than the one experienced by the connection of the STA n.2.

Finally, the average cumulative goodput with VBN (5.3 Mbps) is lower than the one obtained without VBN (5.6 Mbps). As a matter of fact, in both cases none segment and none ACK is lost, but with VBN we are lightly reducing the capacity of the communicative path.

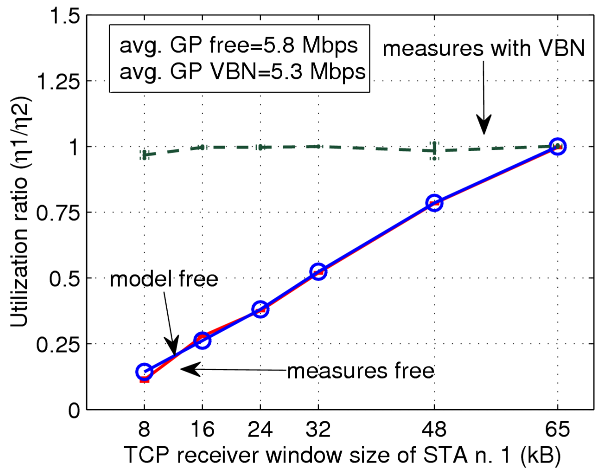

Figure 43: Utilization ratio η1/η2 versus the size of the TCP receiver window of STA n. 1, other STAs have TCP receiver window equal to 65 kB (i.e. 42 pkts). All STAs have only one upstream connection and the physical transmission rate is 11 Mbps

STAs 2÷5 with TCP receiver window equals to 42 packets (65 kB)

The results obtained in this case are reported in Figure 43. Without VBN, all the STAs present an empty buffer, since none STA overcomes the backlogging threshold. Therefore, the RTT of the connections is the same, none segment loss occurs and the ratio η_1/η_2 is equal to the ratio of the receiver windows W_1/W_2.

The average cumulative goodput with VBN (5.3 Mbps) is lower than the one obtained without VBN (5.8 Mbps). This is due to the fact that in both cases none segment is lost, but with VBN we are lightly reducing the capacity of the communicative path. In addition, without VBN ACK losses occur in the AP buffer and wireless interface is better utilized.

2.6.2.5 Coexistence of upstream and downstream TCP connections

This scenario is the same as in other authors [3] and we [69][75] dealt it in previous works. There are a variable number of STAs. Each STA has one upstream and one downstream connection. The TCP receiver window is 42 packets (65 kB) and the physical transmission rate is 11 Mbps.

In this case, the well-know upstream-downstream unfairness issue shows up. To better focus on the specific unfairness issue, in Figure 44 we report the *cumulative utilization factor* of all the downstream connections and of all the upstream connections. The cumulative utilization factor in downstream direction is the sum of the goodputs that all STAs obtain through their downstream connections divided by the overall WLAN goodput. A dual definition subsists for the upstream direction.

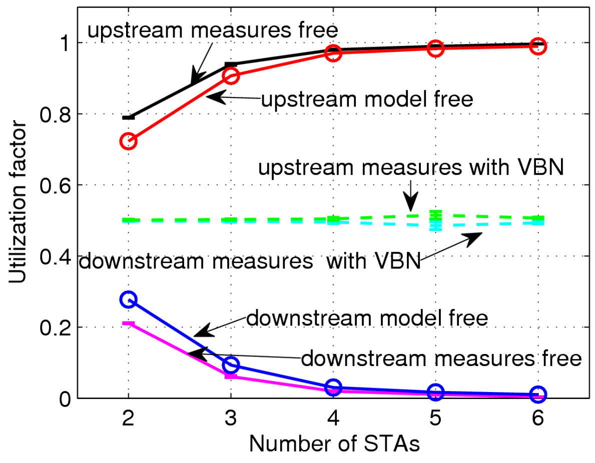

Figure 44: Cumulative utilization factor of upstream and downstream connections versus the number of STAs. Each STA has one upstream and one downstream connection with TCP receiver window of 65 kB and the physical transmission rate is 11 Mbps

In absence of VBN, increasing the number of STAs, the upstream connections starve the downstream connections. In fact, downstream connections experience segments losses in the AP buffer while the upstream connections continue to work in absence of segment loss.

In presence of VBN, the whole system behaviour is lossless. Hence, downstream and upstream connections have the same cumulative utilization factor; i.e., 0.5.

We observe that with a small number of STAs, our model overestimates the performance of the downstream connections. This is due to the fact that, in the formula of the average congestion window *Avg_cwnd*(*p*,*W*,*2*) (see definition d19), we are assuming the retransmission timeout (T_0) as equal to the RTT, whereas the retransmission timeout estimation is practically greater than RTT.

Finally, in presence and in absence of VBN the measured overall average goodput is about 5.3 Mbps.

2.6.2.6 UDP performance in presence of TCP connections

In this Section we analyze the QoS that is perceived by a bidirectional UDP connection in presence of TCP streams. The analysis is carried out only by means of measures since the model developed in Section 2.3 copes with TCP.

The UDP bidirectional connection is bound with the STA n.1. On both sides (i.e., Fixed host and STA n.1) the UDP connections are loaded with a CBR source that sends IP packets 45 bytes long with a bit-rate of 16 kbps; CBR source may be resemble a VoIP source.

The other STAs have only one TCP connection. We in particular consider two cases: only upstream and only downstream TCP connections. Moreover, we vary the number of these STAs.

The performance parameters of the UDP connection that we monitor are the packet loss probability of the downstream and upstream UDP flow, and the round trip time experienced by the bidirectional UDP connection.

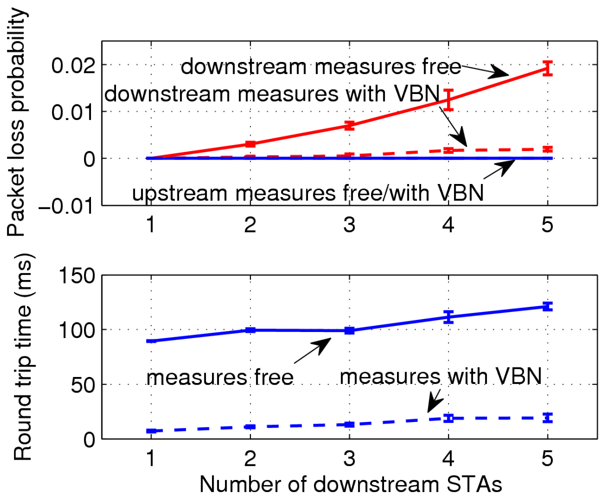

Figure 45: Packet loss probability and round trip time of a bidirectional udp connection bound with STA n.1 versus the number of other STAs that have a single TCP downstream connection with receiver window equal to 65 kB (i.e. 42 pkts). Transmission rate is 11 Mb

Figure 45 reports the UDP performance in case of only downstream TCP connections.

Without VBN, the round trip time is around 100 ms. We point out that, in a more general scenario, this value can be considered as the amount of delay that is introduced by the access network; indeed, in our testbed the fixed network delay is negligible. As far as the packet loss probability is concerned, the upstream UDP flow does not lose packet, whereas the downstream UDP flow loses packets in the AP (although a limited percentage).

Still in Figure 45, we observe that the VBN is able to strongly limit the round trip time and the packet loss probability of both UDP flows, in all the considered conditions.

Overall we conclude that in presence of only downstream TCP connections the performance of a VoIP call may be satisfactory also in absence of VBN. This is due to the fact that TCP downstream connections are losing packets in AP, therefore they do not completely fulfill the AP buffer. The free buffer space is exploited by the VoIP packets. The presence of VBN only increases the performance but its presence is not vital.

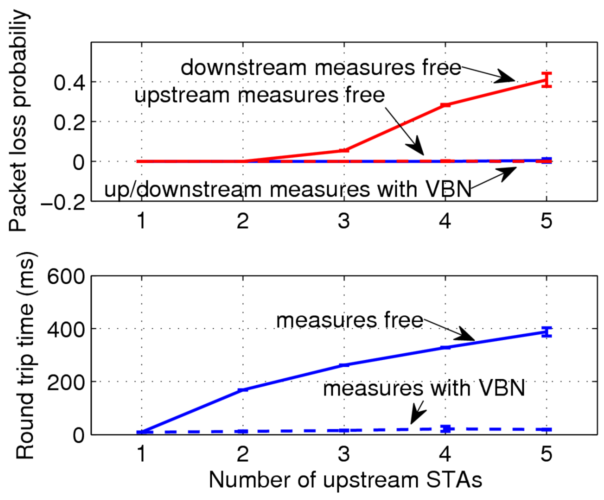

Figure 46: Packet loss probability and round trip time of a bidirectional udp connection bound with STA n.1 versus the number of other STAs that have a single TCP upstream connection with receiver window equal to 65 kB (i.e. 42 pkts). Transmission rate is 11 Mbps

Figure 46 reports the UDP performance in scenarios with only upstream TCP connections. Without VBN, for a number of connections greater than two VoIP calls are impossible due to very high delays.

Still in Figure 46, we observe that the VBN is able to strongly limit the round trip time and the packet loss probability of both UDP flows in all the considered conditions.

Overall we conclude that in this scenario the presence of VBN is vital.

2.7 Conclusions

In a Wi-Fi Hot Spot the interaction of the TCP with the MAC layer implies that the higher the number of TCP segments introduced by a STA in the network, the higher the perceived goodput. This effect might be limited by queuing phenomena on the STA buffer that might occur in presence of upstream connections on the STA.

The number of segments in the network is controlled by the TCP congestion control. For this reason, the upstream connections may seriously degrade the downstream ones since the upstream connections do not experience segment loss as it occurs for the downstream connections. In addition, the upstream connections may also seriously degrade VoIP communications. Practically, such a degradation occurs only when the OSs of the a STA use large receiver window.

The CFTC presented in this Chapter puts the key of traffic control in the hand of network provider.

As a possible extension of this work, we will port the VBN implementation on OpenWRT Linux distribution for embedded devices and we will also include the design and the implementation of the WLAN Capacity estimator.

3 Application Layer: SIP based solution for Wireless Inter-provider Roaming

Deployment of 802.11 Wireless LANs is increasingly on the rise leading to new service scenarios in which users are connected everywhere – everytime. However the IEEE 802.11 standard was designed for short range wireless data transmissions and does not natively provide any support for roaming amongst different access networks. In the more general case, a mobile user should be expected to be able to roam into a visited domain and gain access to the network on the basis of some credentials shared with his home domain or WISP (Wireless Internet Service Provider). There are several mechanisms that can be involved in providing such access control and roaming functionality but no any standard has overcome. In this Chapter a new SIP based solution is proposed. SIP-based authentication is provided end-to-end between user-to-network and network-to-network. The proposed solution realizes full proxy-to-proxy authentication at SIP level, enabling dynamic and secure WISP-to-WISP interworking. The proposed solution has been also implemented and successfully tested in a demonstrating testbed.

The Chapter is organized as follow: first we describe the UniWireless framework, a nationwide distributed Open Access testbed that involves different research units collaborating in the TWELVE [38] national project. The Uni-Fy AAA system, used to manage the collection of involved hotspots, is also discussed. The proposed SIP-based solution is introduced as possible extension of the Uni-Fy gate.

3.1 Introduction

In the growingly popular “always on” perspective ubiquitous wireless networks have accustomed us to, a common problem is given by the plethora of not quite interoperable AAA (Authentication, Authorization and Accounting) systems that are being introduced by different vendors and standardization committees.

Network users need to access remote resources and services in the most comfortable way. On one hand, users want to be able to connect to network services everywhere. On the other hand the network must be readily accessible. In wired network, users are recognized and accounted for on the basis of their physical location, inferred by the physical link they are using. In wireless networks, more sophisticated user authentication systems are required. As GSM experience has shown, hardware identification is not sufficient for authentication and managing and secret-key based procedure are needed. With the recent explosion of Wireless 802.11 LANs (WLANs), new concepts are required. A restricted environment, for instance a private organization, might be willing to provide nomadic access to its employees and to grant limited access to visitors. In wide area environments, for instance public hotspots in airports or stations, the WLANs are used to provide connectivity to nomadic users. However, for a user to be able to enter various different networks, he needs different access protocols and credentials, often translating into a number of installed resident programs and certificates, at the expense of transparency and ease of use.

In this Chapter we describe a nationwide testbed that involves different research units that belong to the TWELVE Project. In this project, all hotspots are unified by a framework for users and service management that we call UniWireless. The collection of hotspots is managed by a common authentication system called Uni-Fy, which can be modularly expanded to accept different forms of authentication, in particular, this Chapter will deal with a “lowest common” authentication functionality based on the captive portal technique and accessible by all wireless clients, and with a more transparent SIP-based authentication technique which can co-exist with the former.

The UniWireless framework is used to demonstrate project results and activities, but it is also deployed to grant access to nomadic users that belong to different research units in all hotspots involved in the testbed. Nomadic users can access to network resource from every place involved in the testbed using the same authentication set throughout the participating units. Sensible data such as passwords or private keys are never shared among hotspots.

3.2 Background and state of the art

In this Chapter we focus on two aspects of public wireless LAN, wireless hotspot management and distributed frameworks where nomadic user can access from several spots using the same account.

These topics are closely related, so we try to focus their main features to describe the relationships between them. Public hotspot management is a relatively new subject, so little literature and experience are available on this. Most wireless hotspot are managed by commercial owners with little interest in sharing customers. Some public hotspots, however, are based on the Open Access Network (OAN) philosophy: a horizontally layered network architecture and business model that separates physical access to the network from service provisioning [20]. The pioneering of OAN management has been the StockholmOpen project [21].

The project consists of a WAN connecting wireless and wired access points [22][23]. The structure of the network allows the coexistence of different Internet Service Providers (ISPs) for user authentication and access to global network. Each ISP that joins in the project must connects its own gateway to the OAN infrastructure.

Other OAN based on the StockholmOpen.net idea are deployed in other cities in Europe, North America and Oceania. The OAN philosophy is based on distributed access spots where nomadic users can access remote resources, but it does not imply any lack of control: a user who wants to connect to global network must be authenticated by a remote authenticator trusted by the access system.

3.2.1 Authentication systems and techniques

The authentication mechanism can be based on the exchange of private information from the client to the remote server with different authentication techniques and protocols. The most diffused technique is the “captive portal” solution based on web pages: when a user connects to the network and requests a page, he is redirected to an authentication web page where, through a secure HTTP connection, he is invited to provide authorization tokens, which can be as simple as username and password, up to certificates or fingerprints. Some famous free and open-source authentication system based on “captive portal” solution are WifiDog [24] and NoCat [25].

Recently also commercial solutions based on the same philosophy appeared on the market, like FirstSpot by Pantronsoft (a Windowsbased manager). All of these are software solutions that provide centralized access control and accounting and run on dedicated servers laying behind the physical access network. Recently, “Hotspot-in-a-box” solutions were developed: the authentication procedure is managed by the AP that provides both physical connectivity to backbone and the user authentication. Also in this solution the captive portal solution can be used.

An overview of the access managements techniques would of course be incomplete without mentioning the 802.1x [26] standard and the work done in 802.11i [27] Task Group. 802.1x defines techniques for user authentication (based on EAP, PEAP, TLS, TTLS, etc.) as well as implementation of secure communications (e.g. based on tunneling). Many of these solution can be used in hotspot management system with safe and scalable features. 802.11i is an amendment of the 802.11 standard that specifies improvement of secure mechanism for wireless network. The standard supersedes the previous security procedure called Wired Equivalent Privacy (WEP) which suffers of security weaknesses. 802.11i is a superset of features introduced by WPA (Wi-fi Protected Access) proposed by the Wi-Fi Alliance and is known as WPA2. It proposes an architecture that includes 802.1x authentication mechanisms, stronger block cipher, encryption protocol, and a 4-way handshake for authentication procedures.

3.2.2 Distributed access framework

Remote resource access management has been a hot topic in the last years. Some architectures based on user authentication to access web-based resources are related to the problem described in the article, however they do not implement an authentication and authorization system. Shibboleth [28] is an architecture that enables organizations to manage a network that allows users to access web resources. The architecture of Shibboleth defines how information must be exchanged between an organization and a provider of digital resources. All the organizations that use this system must previously join a federation. Athens [29] is another access management system to control access to remote resources and services. This management system allows access to protected resources with authentication based on Shibboleth.

Two more project must be mentioned: IRAP and EDUROAM. IRAP [30] (International Roaming Access Protocol) specifies standard interfaces for exchanging authentication, accounting, and management interfaces between providers of public WLAN roaming. It also defines protocols to integrate WLAN with mobile phone network as GSM (2G and 3G), GPRS, UMTS, and cdma2000.

EDUROAM [31] (Education Roaming) is a framework for inter-institution roaming. The system use a RADIUS-based infrastructure and 802.1X protocol to support roaming. In Europe several countries connect to Eduroam project and also non-European countries are members (Australia and Taiwan).

The project involves research and education networks of several countries. The structure supports authentication and roaming is based on a RADIUS hierarchy and the European root is provided by TERENA (Trans-European Research and Educational Networking Association). The two servers act as roots are operated in Netherlands and in Denmark. The structure of the framework is an OAN.

A user who wants to access remote resources from a network must be authenticated. All the traffic in a internal network flows through a programmable router called PAC (Public Access Control) and only authorized clients can access the external network. Authentication procedures are based using VPN, 802.1X and web-based solutions. The deployed solution is based on RADIUS servers because they can support 802.1X.

3.3 UniWireless: System Philosophy and architecture

In this section we describe the structure of the UniWireless framework and the entities that are involved in the global architecture. First we briefly describe the structure of the authentication system in its basic configuration (an in-depth description of the system can be found in [32]) then we describe the development of a new SIP-based authentication procedure.

We need to clarify that a single hotspot involved in the UniWireless can be classified two categories:

Connectivity provider: an entity system to grant network connectivity

Authentication provider: an entity users that must be inquired to

Obviously, a hotspot can simultaneously be a Connectivity Provider and an Authentication Provider.

3.3.1 Uni-Fy gate

Uni-Fy gate is a wireless hotspot management tool developed in the WILMA project under the name "WilmaGate" and now maintained by the TWELVE Project. The TWELVE team added functionalities and extended flexibility to the authentication system to support the UniWireless system.

The Uni-Fy gate is a tool developed in C++ as a collection of user-space application and its modular design is intended for easy addition of new features.

The main feature of the Uni-Fy gate system are:

- support of multiple external authentication providers;
- support of several authentication techniques;
- firewalling;
- accounting.

As shown in Figure 47, the system is made up of two components, Gateway and Gatekeeper, and each component is composed of several C++ modules.

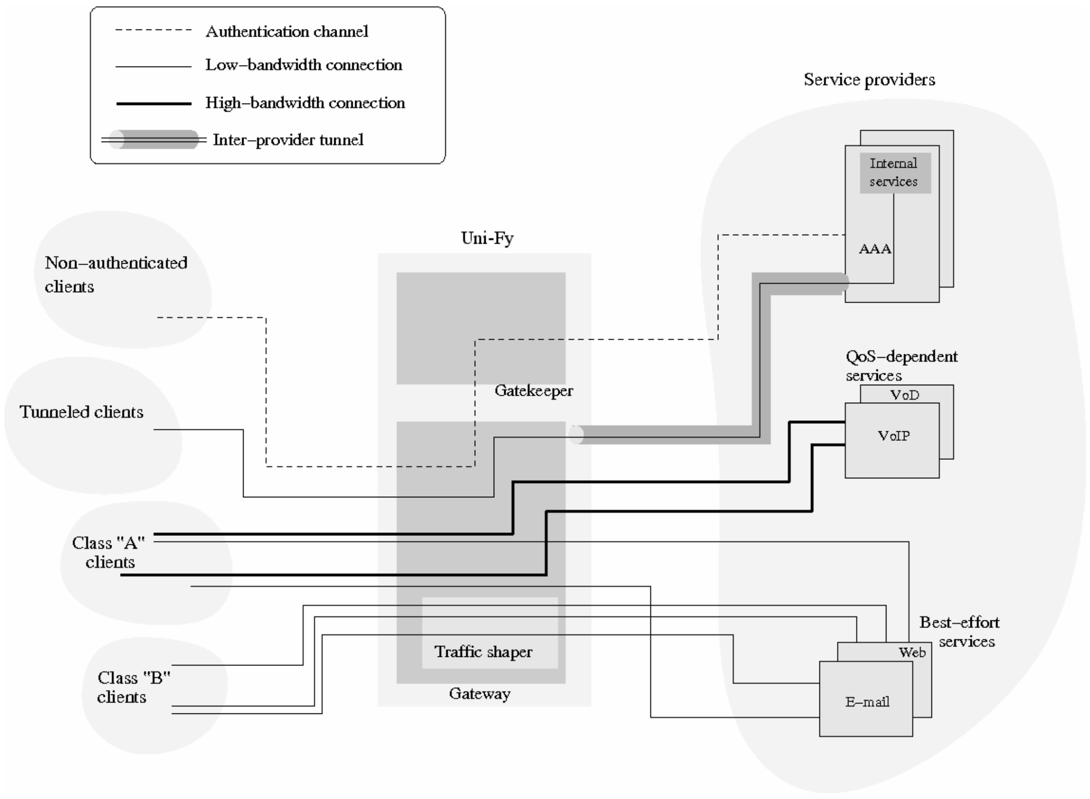

Figure 47 - High level architecture of the Uni-Fy system

3.3.1.1.1 Gateway

The Gateway component runs on a machine with at least two network interfaces and operates as a configurable Layer-3 switch. The component contains a firewalling rules table where, among other data, all (IP address, MAC address) pairs that belong to authorized user are stored. The Gateway receives all the packets from the internal network and manages them. Routing decision are taken according to the firewall rules table:

- if the packet's source addresses belong to an authorized client, the packet is forwarded to the external network;
- otherwise, the packet is forwarded to Gatekeeper component.

The list of authorized client and the packet management policies are kept as simple as possible to avoid computational bottlenecks.

3.3.1.1.2 Gatekeeper

The Gatekeeper component performs all tasks that require more computational processing than a bare packet inspection. It receives all the packets that are not managed by the Gateway through a channel between the two components. The Gatekeeper performs both DHCP management and client authorization based on information received from remote authentication provider. Management of DHCP packets can be performed locally using a built-in DHCP server that implements an appropriate subset of features of a real DHCP server or can be done with interaction of external DHCP server.

The authentication procedure is managed through interaction with a remote authentication system. At this time, the Uni-Fy gate supports two authentication mechanisms, a captive portal technique and a SIP-based authentication method. A successful authentication procedure (carried out in any of the two methods, depending on the user's needs and capabilities) causes an update of the authorized client list both in Gatekeeper and in the Gateway. When a user is authenticated, his traffic is forwarded by the Gateway from the internal LAN to the external LAN and the authorization is automatically renewed from time to time. An authorization can be revoked with an explicit client logout or when a preset authorization period expires with no renewal requests. An in-depth description of supported authentication method follows.

3.3.1.2 Authentication Server

An authentication database containing a list of known users and is inquired by an authentication system to check the credentials that a user provides to access to network resources. Such database can be an institutional RADIUS or LDAP directory as well as an ad-hoc list. An agreement between remote authentication systems and Uni-Fy-driven hotspots must be enacted.

The authentication procedure is always started by the wireless client, which provides its credentials via a secure channel to its home authentication server, so that Uni-Fy never manages the private information of a user. There are not limits about the protocol used to exchange information between authentication system and authentication server, as long as the appropriate firewalling rules are set in the Uni-Fy gateway.

3.3.1.3 Authentication procedure: Captive portal

When a client connects to a WLAN managed by Uni-Fy gate, it receives an IP address following its DHCP request. With this procedure, the client enters the WLAN, but its status is still "unauthorized", so it cannot access the external LAN. Moving the status to "authenticated" is the purpose of the subsequent packet exchange. In order to obtain an authorization, the client contacts a trusted remote authentication server and exchanges information about his identity.

Related to authentication procedure, the exchange of information for authorization includes login and password, cryptographic challenges, secure connection or other techniques. Because the end-to-end feature of procedure, the Uni-Fy gate does not manage private information: it remains transparent to the authentication procedures.

The system allows unauthorized users to contact external trusted authentication servers, but it is also able to limit the traffic to avoid flooding attacks.

Renewal procedures are based on a stateful protocol and they can require light software runs in the client or can be based on standard application (e.g. "captive portal" approach described below). The first authentication procedure supported by Uni-Fy is a web-based solution, called "captive portal".

When an unauthorized user wants to access an external network, he needs to open a browser and requests a web page. The Gatekeeper component intercepts the query and redirects the user's browser to a local page where the user can chose one of the trusted authentication provider. The choice of an authentication provider establishes a secure HTTP connection between the client and the remote authentication server to exchange personal information for the authentication.

A successful authorization procedure generates an update of the client list in the Uni-Fy system and forces the opening of a pop-up window in the client. The pop-up windows must be kept alive along the whole session because renewal is based on its periodic refresh.

Although simple and flexible, this method can sometime suffer of some security limitations. Moreover, since it is based on HTTP interface, it is not very tight to emerging real-time multimedia applications.

In the next Section a new SIP based solution is introduced. SIP-based authentication is provided end-to-end between user-to-network and network-to-network (in case of different ISPs are involved). The proposed solution does extend the base SIP authentication procedure by realizing full proxy-to-proxy authentication at SIP level, enabling dynamic and secure WISP-to-WISP interworking.

3.3.2 A novel SIP based authentication procedure

In this Section we will see a practical demonstration of the feasibility of the integration within the Uni-Fy gate of different authentication mechanisms. In particular, we will see a secure authentication scheme based on Session Initiation Protocol (SIP) [33], related to the scheme that is used in the 3GPP/IMS security framework [35].

The basic idea is that SIP-based authentication mechanism can interact with the authentication system improving the authentication capacities: devices like Wi-Fi phones or devices without display for web-browsing will be able to access to remote resources. The SIP based mechanism can coexist with the captive portal mechanism.

3.3.2.1 SIP Authentication overview

SIP, the Session Initiation Protocol, is a signaling protocol for Internet conferencing, telephony, presence, events notification and instant messaging. SIP was developed within the IETF MMUSIC (Multiparty Multimedia Session Control) working group, with work proceeding since September 1999 in the IETF SIP working group. It is specified in RFC 3261 [33], and successive related RFCs.

SIP authentication model is inherited from the HTTP digest authentication (RFC 2617 [34]). Originally also the HTTP basic authentication was supported by SIP, but it has been deprecated by the RFC 3261, due to the insecurity on sending the shared secret (i.e. the users' password) in clear within the request messages.

3.3.2.1.1 Digest Authentication

Digest authentication follows a challenge-response scheme based on a shared secret key (password). In a SIP-based network, the authentication can take place between a user agent and a server (e.g. a registrar server or an intermediate proxy).

A User Agent (UA), in the following also referred as users' terminal, represents one of the communication's endpoints. It can be a hardware or software device with the capability of initiating or receiving a call based on SIP signaling. A terminal can work either as a server (UAS), if it receives the call, or as a client (UAC) if it initiates the call. Authentication is normally requested by the UAS (to be sure on the UAC identity before processing its requests) or also by the UAC (in such case, mutual authentication is provided). SIP Digest Authentication procedure is based on a four-messages exchange. Figure 48 illustrates a successful authentication procedure.

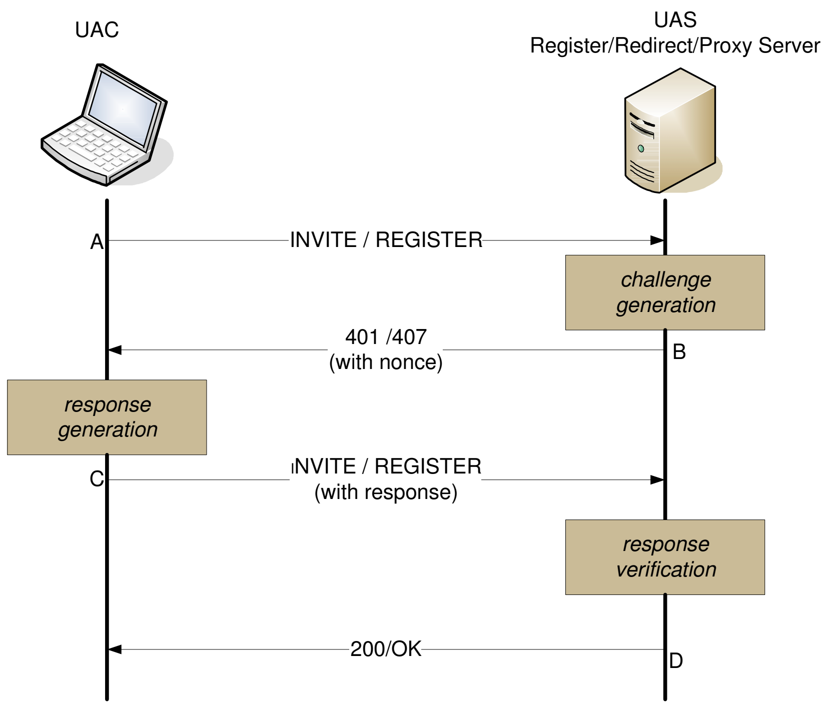

Figure 48 - SIP Digest Authentication Scheme

The first of these messages (A) is sent by the UAC, and it's a SIP request. Usually, this request is an INVITE message (if the UAC is trying to establish a new session) or a REGISTER message (if updating the location information). Often this first request doesn't include any authentication information nor credentials, since the other side has not challenged the UAC yet. The request is received by a UAS, that can be a registrar, a redirect or a proxy server. A registrar is a server that accepts REGISTER requests and uses the information provided by successful registration requests to update location information within a particular domain it is responsible for [33]. A redirect server is a UAS that generates 3xx responses to requests it receives, redirecting the UAC to contact an alternate set of URIs [33]. A proxy server receives SIP requests and forwards them on behalf of the requestor [33]. After generating a challenge for the UAC, these entities send the second message of the process (message B), containing the challenge for the UAC. The challenge is composed by different parameter such as the realm, authentication method, algorithm, and nonce. This response message will be either an *Unauthorized* response message containing a WWW-Authenticate header if sent by UAS, registrar or redirect server (401 response code), or a *Proxy Authentication Required* response message containing a Proxy-Authenticate header if sent by a proxy server (407 response code).

Once the UAC receives the challenge it calculates the response. This response is computed using the algorithm specified within the request (usually the MD5) with parameters such as nonce, the shared secret key, user-name and some others [33]. Once the response to the challenge has been computed, a new SIP request (message C, that is the same method of message A), is sent by the UAC, including now a response parameter, which is actually the response to the challenge received within the previous message.

After receiving this new request, the UAS checks if the received challenge response equals the expected value (i.e. the value obtained by calculating response by means of the locally stored user's shared secret). If it succeeds, the UAC is successfully authenticated and the proper response message corresponding to the UAC request is sent by the UAS (message D).

3.3.2.1.2 AKA Authentication

The previous procedure such as defined in the RFC 3261 is referred as SIP Digest authentication procedure.

The authentication framework has been also extended by the Third Generation Partnership Project (3GPP) in order to be use and interoperate with the 3G systems. The authentication mechanism used in 3G networks is the Authentication and Key

Agreement procedure (AKA). AKA is basically a challenge/response authentication mechanism that naturally provides mutual authentication (user-to-network and network-to-user), and roaming facilities. In order to let the SIP-based signaling platform to inter-operate with 3G systems the SIP AKA [36] authentication method has been defined. In our proposal we, as it will be described in the following sections, we use AKA as base authentication mechanism also for WLAN-based access systems. The main benefit of such approach is obviously the reusability of the 3GPP infrastructure and the compatibility between the new WLAN access networks and already development 3G systems.

Authentication in 3G lays on two different keys called K_U and K_I that are stored both in the client's SIM card and in the Home Subscriber Server (HSS). The SIM card consists of two virtually-different modules: USIM (UMTS-SIM), containing K_U, and ISIM (IP multimedia SIM), containing K_I.

As already explained above, AKA is a challenge-response mechanism that provides both mutual authentication and session keys generation. AKA is used both to authenticate the radio network as well as the IP services associated to 3G (IP Multimedia Subsystem - IMS) [37]. The authentication procedure in 3G is represented in Figure 49

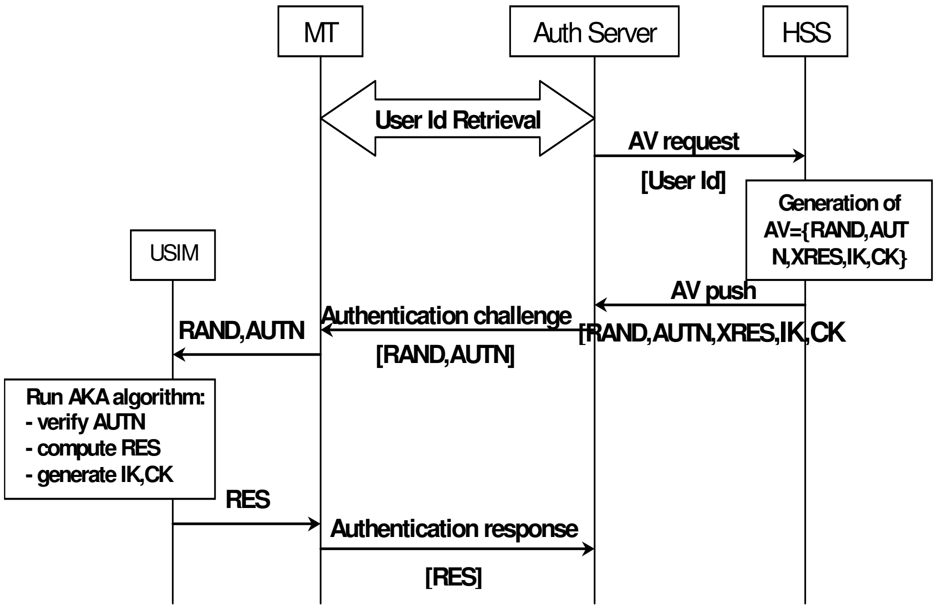

Figure 49 - AKA successful procedure

When a node on behalf of the 3G network wants to authenticate a mobile user it use a set of authentication information called Authentication Vector (AV). An AV consists of a set of parameters required to achieve a successful authentication procedure, and is provided by the HSS based on the secret key and on a sequence number SQN.

An AV is composed of: RAND, a random challenge called nonce when talking about the Digest procedure; AUTN, a token used to authenticate the network towards the client; XRES, a expected result (the expected response to the challenge from the client); IK, a session key for integrity check; and CK, a key used for encryption that differs each time an authentication procedure is required.

When requested, a new AV is passed to the authentication entity/node (the authenticator) in the network and is used to challenge a new supplicant MT. Once the authentication request arrives to the USIM module of the MT, the USIM uses the secret key K and the sequence number SQN to verify the AUTN. If both AUTN and SQN are valid, the network has been authenticated and the user-to-network authentication procedure proceeds. The USIM module calculates the keys IK and CK (that will remain local, this means they will not be sent to the network) as well as the authentication result (RES). RES is sent to the authenticator to be compared with XRES. If they match, the MT/user is authenticated and further communications will be protected with IK and CK keys.

3.3.2.1.3 DIGEST – AKA Authentication

This AKA mechanism can be combined with the Digest one, originating the *Digest AKA authentication* scheme [35]. One of the main feature of this mechanism is that it lets the UAC check the identity of the network; this is done bye UAC extracting the AKA RAND and AUTN parameter values from the nonce parameter of the proper authenticate header field.

This parameter is formed as base64 encoding of RAND, AUTN, and server-data, i.e

```
nonce=base64(RAND,AUTN,server-data)
```

where server-data is some optional data provided by the UAS. As already explained, the supplicant (i.e. the 3G MT and/or the SIP UAC) uses those parameters to authenticate the network.

After extracting the network authentication token, AUTN, and checking that it could only come from a valid network, the UAC proceeds to calculate the answer to the challenge (actually the RAND value) based on the shared secret (K). Another main benefits of the AKA mechanism is that it generates a sort of password for every authentication procedure. This password is the RES value and it is provided by the HHS within the AV and computed separately by the UAC and by the HHS based on RAND and K values (RES=f(RAND,K)).

This RES parameter is used to form the digest response parameter. Such response parameter can be calculated for example as MD5 of a set of authentication parameters as

```
digest-resp=MD5(MD5(username:realm:f(RAND,K)):
          :base64(RAND,AUTN,sd):MD5(A2))
```

where the operator ":" just indicates a concatenation; f(RAND,K) is the unique password for each request; sd is the server data described above and A2 is a string whose value is depends on the value of a parameter called qop (refer to [33] for further details). The whole authentication process is shown in Figure 50.

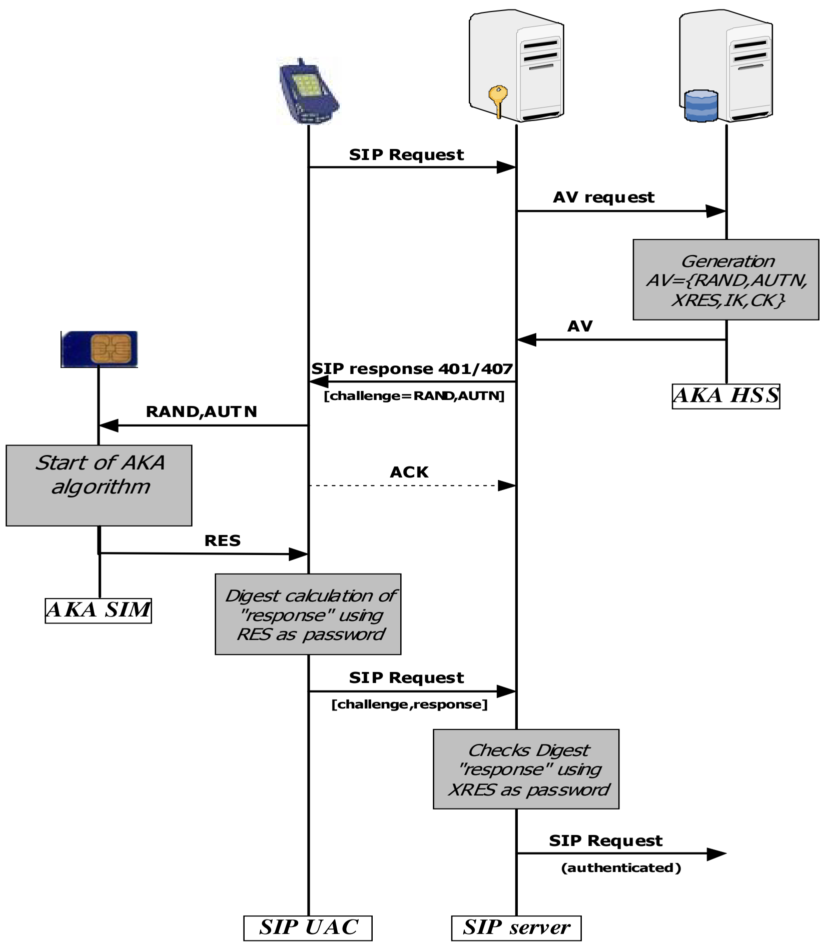

Figure 50 - AKA Digest Authentication Scheme

3.3.2.2 SIP-based authentication method

In terms of inter-ISP roaming, the relation between the ISP providing network access (the Access Provider) and the ISP providing authentication functionality (the Authentication Provider) can be one of the following:

1. The Authentication Provider can be a different administratively entity, separate from the Access Provider, with a strict trust relationship with it; different Access Providers can relay one the same centralized Authentication Provider; such scenario can be further extended by a centralized hierarchical authentication architecture;
2. Each Access Provider implements a local Authentication Provider that shares with the other trusted Access Providers; in this case the various ISP (acting as both Access Provider and Authentication Provider) will form a kind of web of trusted-ISP; the users when accessing through an Access Provider can choose their own Authentication Provider with which try to authenticate.

The proposed idea is to realize such web of trusted-ISP by means of the same signaling platform used for multimedia real-time service and used by 3G mobile networks.

When a mobile user roams into a new visited network it try to register with his/her own SIP registrar server (acting as home registrar or *Home Authentication Provider*). Such procedure is intercepted by the local Gatekeeper (the access controller) administrated by the visited ISP and redirected to the Home Authentication Provider opportunely modified with ISP-to-ISP authentication and authorization capabilities, according to the architecture described in the previous sub-Section.

In order to assure ISP-to-ISP authentication and correct authorization information retrieval from the Home Authentication Provider (i.e. the remote SIP registrar server), an extension of the standard UAC-to-UAS SIP authentication procedure is proposed and has been implemented.

Two new header fields allowing authentication between two intermediate SIP entities are here defined: Proxy-To-Proxy-Authenticate (shortly referred in the following as *pp-authenticate*) and Proxy-To-Proxy-Authorization (shortly referred in the following as *pp-authorization*).

According with the standard SIP authentication procedure, the *pp-authenticate* header is used to carry authentication request information, while *pp-authorization* header is used to carry authentication response information.

The *pp-authenticate* header is used by a generic intermediate proxy to authenticate a next-hop proxy or next-hop UAS, in order to correctly trust information sent as response from such next hop entity. The *pp-authenticate* header is inserted by the proxy within a proxing SIP request message, while the *pp-authorization* is inserted in a SIP response message by the next hop entity in response to the *pp-authenticate* request.

The authentication method used with the *pp-authenticate* and *pp-authorization* can be anyone of the already proposed authentication methods in SIP, without any restriction, and is selected by the intermediate node that starts the proxy-to-proxy authentication procedure.

Let us now consider how the new SIP extension applies to a SIP-based access control scenario. Let us consider a mobile user that roams into a new visited network and tries to register his/her UA with his/her own home registrar server within the home network/ISP. Let us consider the case in which there is a trust relationship (with proper authorization and roaming policy) already established between the home ISP (acting as Authentication Provider) and the visited ISP (acting as Access Provider). The register request sent by the UA is then intercepted by the gatekeeper in the visited network and forwarded to the user's registrar server.

Figure 51 shows the complete registration and authentication procedure exchanged between the user and the Access Provider and between the Access Provider and the Authentication Provider.

When the registrar server receives the first register request sent by the UA it starts standard UAC-to-UAS authentication procedure, by sending a 401 Unauthorized response message containing a WWW-Authenticate header with the authentication method (AKA is expected to be used) and the challenge, as described in Section 3.3.2.1.1 and 3.3.2.1.3. The message is transparently forwarded to the UA.

When receiving this response the UA sends a new register request populated with an Authorization header with the proper authentication challenge response. When intercepting this authenticated register request, the gatekeeper starts a new proxy-to-proxy authentication procedure attempting to challenge the remote registrar. In the register request a new *pp-authenticate* header field is added with realm, algorithm, username, nonce, method, uri and other parameters according to the selected authentication method used for the proxy-to-proxy authentication.

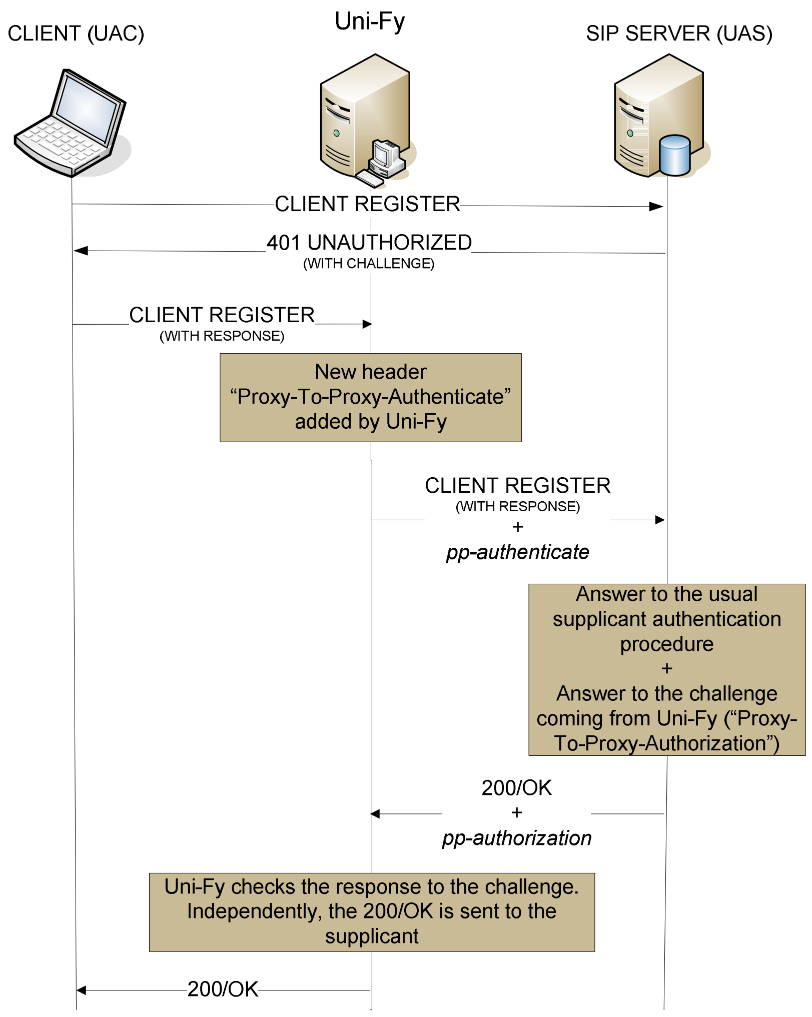
CLIENT (UAC)
Uni-Fy
SIP SERVER (UAS)
CLIENT REGISTER
401 UNAUTHORIZED
(WITH CHALLENGE)
CLIENT REGISTER
(WITH RESPONSE)
New header "Proxy-To-Proxy-Authenticate" added by Uni-Fy
CLIENT REGISTER
(WITH RESPONSE)
+
pp-authenticate
Answer to the usual supplicant authentication procedure + Answer to the challenge coming from Uni-Fy ("Proxy-To-Proxy-Authorization")
200/OK
+
pp-authorization
Uni-Fy checks the response to the challenge. Independently, the 200/OK is sent to the supplicant
200/OK

Figure 51 - Proposed Authentication Scheme

Although any authentication method can be used, in the rest of this section a Digest authentication is supposed to be used.

The receiving registrar server (the UAS), according to this procedure processes both the *Authorization* and the *pp-authentication* header fields (the former for user authentication, the latter for proxy-to-proxy authentication). For the latter, a new *pp-authorization* header field is added into the registration response generated after the user authentication has been performed (a 200 OK response is sent in case the authentication process succeeded).

This *pp-authorization* header field should include, at least, the computed response to the challenge sent, together with the other parameters sent with the *pp-authenticate* header field.

Once the gatekeeper receives such message with the *pp-authorization* header (the fourth message) it checks if the new response match with expected result that is locally calculated based on the secret shared between the Access Provider and the remote Authentication Provider (i.e. the registrar server). If it succeeds, and if the response code sent to the UAC from the UAS is a 200/OK code, the Gatekeeper updates its authorization table, changing the status of the new user/terminal to "AUTHORIZED". If the authentication verification fails, the status is changed to "FORBIDDEN". Figure 52 represents this operation.

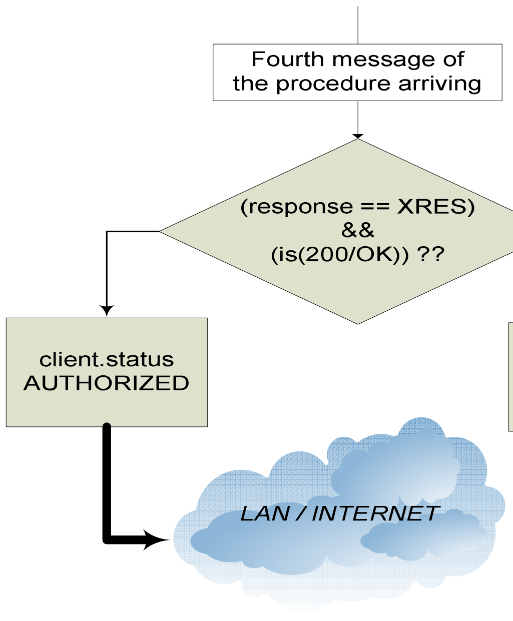
Fourth message of the procedure arriving
(response == XRES) && (is(200/OK)) ??
client.status AUTHORIZED
client.status FORBIDDEN
LAN / INTERNET

Figure 52 - Client's state transaction in Uni-Fy

The whole authentication and authorization scenario has been implemented in a testbed, based on the Uni-Fy access control mechanism described in Section 3.3.1. The Gateway/Gatekeeper nodes have been realized based on the Uni-Fy implementation provided in the TWELVE project framework. The Gatekeeper plug-in for SIP has been developed in C++ based on the reSIProcate C++ SIP stack library [39].

Finally, the registrar server, opportunely extended with proxy-to-proxy authentication has been implemented in Java, based on the mjsip SIP stack library [40].

3.3.2.3 Security considerations

A deep analysis of all possible security threats on the proposed access solution is out of the scope of this work and will be the objective of further works. However some considerations are hereafter discussed.

As far as a strong security mechanism is not implemented within the access network at layer 2 (e.g. 802.11i with AES) or at layer 3 with IPSec, different attacks can be mounted against both user data confidentiality and access control. For example, as far as access control at data plane is performed on the basis MAC/IP address matching, such service can be attacked by mounting a DoS attack toward a legitimated terminal and by reusing the same MAC/IP addresses. The use of short re-authentication timeouts aims to mitigate such type of attacks.

Regarding the proposed SIP authentication procedure based on the new Proxy-to-Proxy-authentication, the same considerations and limits stood out for the standard UAC-to-UAS authentication and described in RFC 2631 (Section 23) are still valid.

3.4 UniWireless implementation

The UniWireless framework is a collection of hotspots providing network connectivity in several places. All the universities joining the TWELVE project installed a Uni-Fy gate and some of them act also as authentication provider. Each university decided autonomously how to install the authentication software related to its previous network configuration and architecture. In Figure 53 an overview of the topology of the system is shown.

The implementation of the Uni-Fy gate can be done with Gateway and Gatekeeper run on a single machine or in two separated computers. The chosen implementation does not influence the performance of the other authentication systems in the UniWireless framework: among Uni-Fy gates there are no interactions. Each system work separately and only manages only users in the private LAN that it controls. The interactions in the system are among the Uni-Fy gate and the remote authentication servers. In the set-up phase of the Uni-Fy gate a list of the remote authentication server must be inserted. A remote authentication server is defined by IP address, domain and information how to reach authentication procedure. The authentication server must be trusted, so previous agreement must be subscribed between connectivity provider and authentication provider.

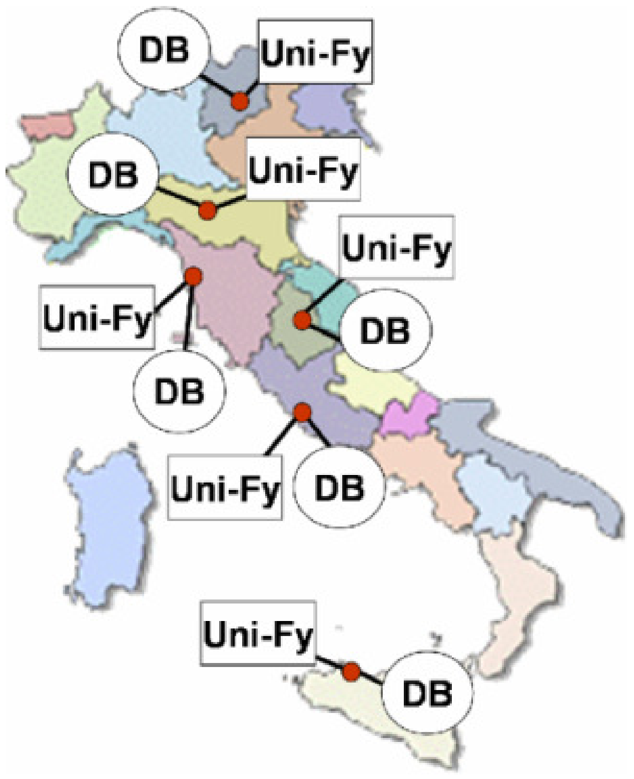

Figure 53 - Overview of the UniWireless framework

After a correct authentication procedure the user receives an acknowledgment from the remote authentication server and has rights to access remote resources. The remote authenticator must send also information to the Uni-Fy that manages the LAN where the user is to confirm the change of the user state from unauthorized to authorized. The interconnection among Uni-Fy gate and database does not require a dedicated path. In the current configuration the global LAN is used.

The first step of the building of the system defines a structure where the authentication procedure was a web-based solution. The testing and development of a new authentication procedure in a HotSpot did not require changes in the other HotSpot involved in the Uni-Wireless.

In general the framework can be used to test new algorithms, authentication procedures and protocols without impacts on the other authenticator. Obviously if the new version of Uni-Fy gate that provides SIP-based authentication is implemented only in a HotSpot, it is a limit for nomadic user that need to use this procedure to access to the network. To provide this new authentication procedure is supported by all the entities involved in the framework, an update of the Uni-Fy system is required. It is important to underline that updating of the authentication system cause only a re-compiling of the system and (in some case) adding new configuration parameters. It does not force to install new server that support locally the new authentication procedure. With updating of Uni-Fy gate a nomadic user can access from everywhere using his own credential provided by a remote authenticator.

4 Application Layer: SIP based solution for mobility management

This Chapter describes a Session Initiation Protocol (SIP) based solution for mobility management that provides seamless mobile multimedia services in a heterogeneous scenario where different radio access technologies are used (802.11/WiFi, Bluetooth, 2.5G/3G networks). The solution relies on the so called "Session Border Controllers" which are now widely used in many commercial SIP telephony solutions, mainly to deal with NAT traversal. Session Border Controller functionality has been extended to support seamless mobility for multimedia applications. A prototype of the proposed solution focused on VoIP services has been implemented in a test bed which is able to perform seamless handovers (and NAT traversal) using the 802.11, Bluetooth and 3G (UMTS) access networks. Measurements results are reported which analyze the performance of the solution in a real world environment, using commercial WiFi and 3G services.

4.1 Introduction

Several different wireless access technologies are now available that can support real time services, in particular voice over IP (VoIP). Currently, these technologies span from cellular networks (UMTS), LAN technologies (802.11 a/b/g) and even PAN technology (Bluetooth), but other technologies will become mature in the very near future (e.g. 802.16). Multi standard terminals, laptops, tablet PCs, PDAs and even phones are now able to use more than one interface at a time. One of the goal of next generation mobile networks (often referred to as 4G network) is the integration and interoperability of different access technologies (including fixed access with "fixed mobile convergence") into a single system or better into a single service presented to the user. The integration can happen at different levels of the protocol stack and various solutions are under study which operate at these different levels. In this work we focus on an application level solution based on the SIP protocol [33], [41]. The advantage of this type of solution is that it easily adapt to whatever underlying access technology is used.

The service scenario that we have considered is *wireless VoIP*. Nothing prevents to extend it to wireless multimedia (e.g. for video communication) provided that the access network(s) offer suitable capacity. Wireless VoIP service can be accessed over: (i) Local Area Networks (LAN) where voice communication are provided to a company over an enterprise wireless LAN (WLAN) or to people accessing public WLAN hot spots; (ii) Personal Area Networks (PAN) for example offering voice communication using Bluetooth technology in a domestic environment or in a small office; (iii) Wide Area Network (WAN) thanks to high data rate 2.5G/3G cellular technologies. The ability to use multiple networks in parallel gives the user a possibility to choose the most economical or the most performing access network at a given time, or conversely it gives the service provider the possibility to use the most suitable connection for each application. Therefore we face the requirement to support seamless mobility on multi-mode terminals, with the ability to place and receive calls over the most suitable wireless interface and to maintain VoIP sessions alive while handing off between the different access networks. As most of the wireless access networks are currently using private IP addresses and are connected through NAT (Network Address Translation) elements, the “NAT traversal” is one of the most critical issues to ubiquitously deploy VoIP services in the real world. The support of the “NAT traversal” combined with mobility and handover functionality is a very important requirement.

In this Chapter we describe a solution for this issue. The solution takes care of the “vertical mobility” of a user among different access networks/technologies, considering that for each different attached network the terminal will receive and use a different IP address. On the other hand, the movement of the user among base stations of the same technology/network (e.g. among different access points in the same WiFi campus, or among different 3G cells in the cellular network of an operator) is handled by specific mechanisms of the given access network and no IP re-configuration is required.

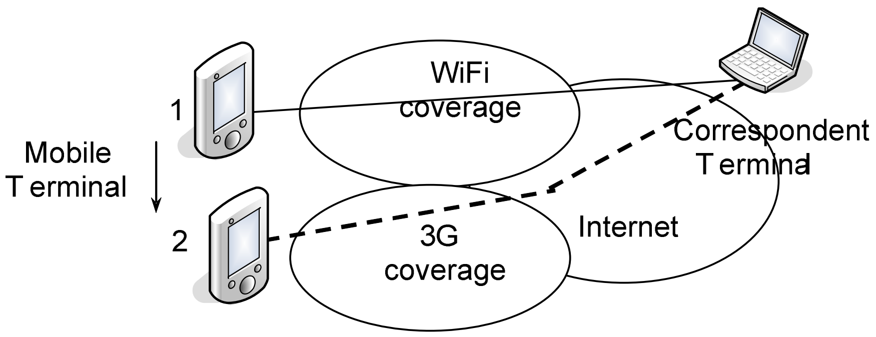

Figure 54 - VoIP handoff scenario in a Wi-Fi/3G interworking scenario

In our solution, we consider a Mobile Terminal (MT) with multiple network interfaces that can be active at the same time making it possible to realize a “soft handover”. As an example Figure 54 illustrates the scenario of a handover between a WiFi and a 3G network (this scenario is a subset of what has been also implemented in our testbed that will be described later on). The Session Initiation Protocol (SIP) will be used for both “traditional” VoIP signaling and for supporting terminal mobility. The solution integrates mechanisms to enable MTs to make and receive VoIP calls regardless if they are located inside a public or private IP network.

The Chapter is organized as follows. We introduce the main concepts of the proposed solution (Section 4.2) and then discuss the details of the signaling procedures (Section 4.3). Section 4.4 describes our testbed and the achieved measurement results, while Section 4.5 provides a report of existing work linked to our solution.

4.2 Proposed Solution

The fundamental concepts of the proposed solution can be illustrated with the help of Figure 55. MTs have access to different networks (in the figure, WiFi, Bluetooth and cellular 3G network), which can overlap their coverage areas. The MT has separate interfaces, each one dynamically receives its (private or public) IP address from the corresponding wireless network. The MT logically contains the User Agent (UA, i.e. the SIP client) and a Mobility Management Client (MMC). The MT uses a Session Border Controller (SBC) [42] to access VoIP services from IP access networks often based on a private IP addressing scheme and operating behind a NAT/FW box. The SBC contains a Mobility Management Server (MMS) which is the main entity controlling the user mobility. Thanks to the interaction between the MMC in the mobile terminal and the MMS in the SBC the device can move between IP subnets, allowing the UA to be reachable for incoming requests and to maintain VoIP sessions across subnet changes. The “CT” node shown in the picture is the Correspondent Terminal that communicates with the MT.

SIP Registrar functionality that are not directly related to handover/mobility management procedures can be performed by an external SIP Registrar, as shown in the figure. Obviously the MMS and SIP Registrar can be implemented in a single element if required.

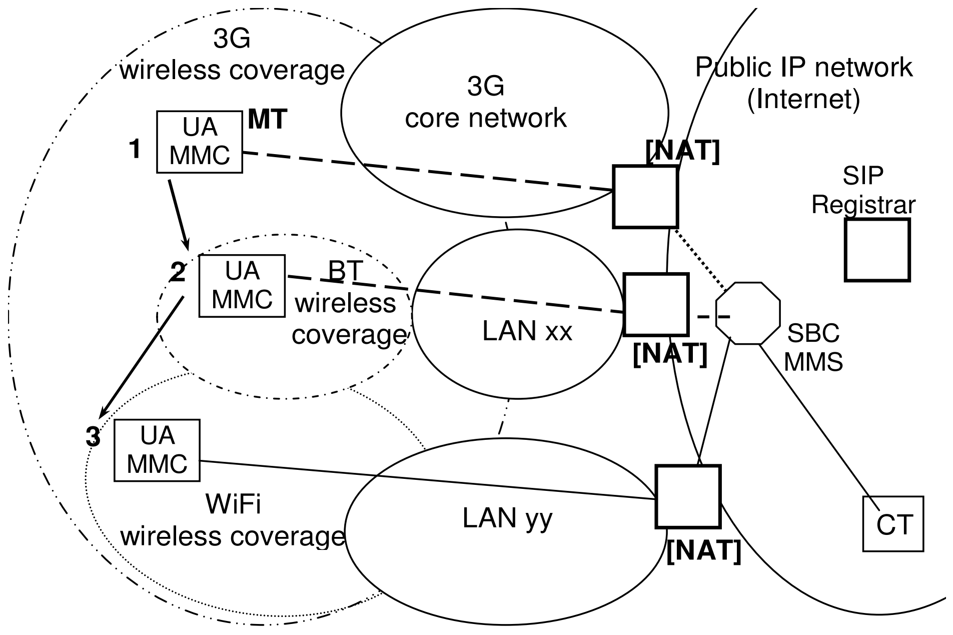

Figure 55 - Architecture

4.2.1 From SIP Session Border Controllers to mobility management server

While a large part of the target terminals are using private IP addresses and are "hidden" behind a NAT, signaling protocols for VoIP, like SIP, do not "natively" support full NAT traversal. NAT traversal mechanisms are needed to allow terminals in "private" networks to be reached (i.e. "invited" to a phone call), and to allow the media streams to be setup between the caller and the called UAs, notwithstanding NAT boxes that could be placed at the borders between the private and the public network. IPv6 promises to overcome the NAT issues but wide spread diffusion of IPv6 is not foreseen in the short/medium term. Moreover, in future IPv6 networks, the problem of address/port mapping may still be present due to some restrictions introduced for security reasons. A SIP Session Border Controller is a session-aware device that manages SIP calls at the border of an IP network. It is aware of both signalling and media flows. An SBC may have several functions, one of the more interesting is solving the problem of NAT/firewall traversal, dealing with different NAT and/or UA behaviors. In this respect, an SBC provides NAT/firewall traversal without additional customer premise equipment, and without the replacement of existing firewalls and NATs. An SBC does not require any additional STUN/TURN node nor STUN/TURN protocol support ([43], [44]), neither at UA nor at SBC side. Besides NAT traversal, the SBC may have several functions, we will only list two of them: (i) the SBC can provide media interworking function for different media-related functionalities such as: media transcoder, media encryption and protection against various media-based attacks; (ii) the SBC can provide signaling and media wiretapping system, which can be used to enforce requests for the lawful interception of communication sessions.

From the point of view of SIP signaling, an SBC can act as a SIP B2BUA (Back-to-Back UA) or as a special SIP proxy. In the former case, the SBC works as an intermediate node that breaks the signaling path between two UAs and interconnects them (e.g. setting up a call) by means of establishing separate end-to-end connections between itself and each remote UA. In the latter case the SBC does not break the signaling path between the two UAs; instead it relays signaling requests and responses between remote UAs and other proxies, operating all SBC-specific function extending the normal proxy behavior as defined by RFC 2361. In addition to what is defined in the SIP standard for the operation of a SIP proxy, the SBC will modify the description of media session contained in the SDP, and some other SIP header fields like for example the Contact header field. Despite this extended behavior, obviously all outgoing signaling remains fully compliant with SIP standards. In our solution, we preferred to use a “proxy like” SBC as it is lighter and more “transparent” with respect to the SIP signaling among the endpoint UAs.

In order to manage the user mobility, we propose to add the MMS element into the SBC. The MMS is an “anchor point” for the media flows which are transmitted over the wireless access networks directed to (and coming from) the MT. When the MMC in the MT detects that a handover is needed, it will request the handover to the MMS (via a SIP message) over the “target” network. Then the MMS (in the SBC) will update its media proxy and will start transmitting and receiving the media over the target network (details are provided in the next Section). Note that the entire handover procedure is handled by the MT and the SBC, letting the CT (and other SIP intermediate nodes) completely unaware of what is occurring. Instead, in the traditional SIP approach the CT is directly involved by the MT with a new “Re-INVITE” transaction. In such case, the CT is in charge of performing the handoff by establishing a new media flow using the IP address of the terminal in the “target” network. A first advantage of the SBC based handover with respect to CT-based handover is that the solution does not rely on the CT capability to perform the handover. Though the SIP standard requires that terminals are able to support Re-INVITE, compatibility issues may arise and it will be difficult to verify that the handover works with all possible SIP terminals. Moreover executing the handover with the CT could lead to high delays, for this reason various solutions have been proposed trying to overcome this problem by introducing intermediate entities as temporary anchor points (see [45], [46]). In our solution the anchor point is not temporary, but permanent. This permanent media relay in the path is not a very efficient solution, but in practice we witness several running services that are implemented in this way. It is also worth noting that the only other solution which allows to overcome a “symmetric NAT” is using TURN [44], but this exactly requires a media proxy box that acts as permanent media relay (and it introduces greater setup and handover latency respect to a SBC-based solution). If we can assume that a Session Border Controller is already in the path the introduction of a media relay is definitely not a shortcoming introduced by our solution. We are simply extending the SIP and media processing capability of the SBC in order to support the handover.

Note that the proposed solution can be applied exactly as described both in networks with private IP addresses and in networks with public IP addresses. In a scenario with private IP networks and in a mixed scenario (where the MT can roam among both networks with private IP addresses and networks with public IP addresses) the proposed solution is able to solve the NAT traversal issues together with the mobility management. If we consider a scenario where the terminal is only roaming among networks with public IP addresses, we loose one of the advantages of our solution (as no NAT traversal is needed). This scenario does not seem realistic in current networks. Anyway, our solution could also prove useful in this context, due to the other advantages listed above: the handover procedure does not rely on the remote terminal, the handover delay could be better controlled. Moreover for VoIP services offered by a public network operator, the need to provide “lawful interception” could imply that media flows are conveyed in any case through a media relay.

For the sake of simplicity, we only describe the solution considering a single centralized SBC. In real life, redundancies must be considered to achieve reliability (e.g. the SBC needs to be replicated). Likewise, the SBC functionality could need to be distributed for scalability reasons. A possible interesting approach is to split the signaling and media relay capability of the SBC (see for example [47]), which allows to deploy a set of media relay boxes that can be placed “close” to the MT.

4.2.2 The Mobility Management Client in the Mobile Terminal

The Mobility Management Client can be implemented as shown in Figure 56 as a separate entity running on the MT that masquerades all mobility and NAT traversal functionality by relaying both signaling and media flows. In this case the SIP User Agent sees the MMC as default “outbound proxy” (which means that the UA will send all SIP message to the MMC) and it has no knowledge of the handovers. Existing SIP UAs can be easily supported/reused without any changes. A different solution would be to integrate the MMC functionality within the UA, which will likely imply a greater efficiency in the use of processing resource of the MT. The two solutions only differ in the internal implementation, while there is no difference in the external behavior of the procedures. In our test-bed we used the first solution (the one depicted in Figure 56) so that we can use any existing SIP User Agent.

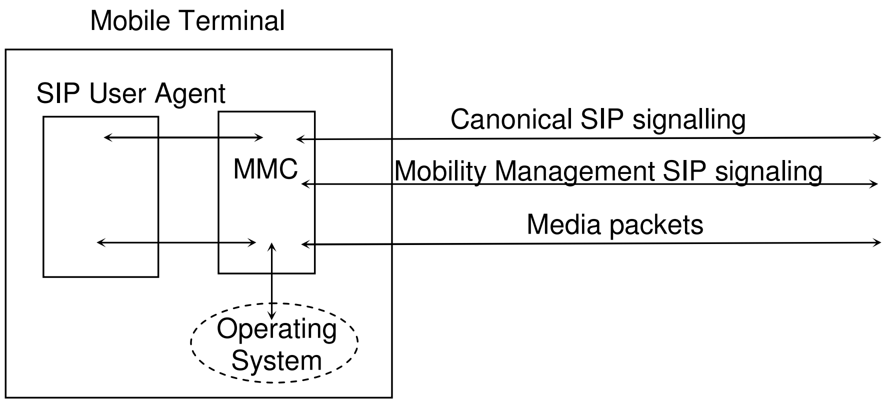

Figure 56 - The MMC in the mobile terminal

In order to configure the IP addresses on the MT interfaces, existing mechanisms are used (e.g. PPP on the 3G interface, DHCP on WiFi LAN and Bluetooth PAN). When multiple interfaces are active, the MMC needs to select the preferred interface for sending/receiving the media flows (while the terminal is involved in a call) or for exchanging SIP signaling (both during calls and in idle state). The choice of the selected interface performed by MMC may depend on cost aspects and/or on QoS issues (signal strength, perceived packet loss and/or delay). The discussion of these criteria are out of the scope of this work.

4.3 Specification of the Procedures

As described in the previous section, the mobility management involves four main functional entities. On the MT sides there are the SIP UA and the MMC, while on the network side there are the SBC with the MMS and a SIP Registrar.

The SBC enhanced with the MMS is needed to manage MT handoffs between different access networks providing service continuity and NAT traversal. The SBC is able to process both SIP protocol header fields and Session Description Protocol (SDP) [48] bodies in order to force itself as relay for the media packets.

In the SIP architecture, the SIP Registrar records the current location(s) of the user. An SBC coupled with an external backend SIP Registrar introduces a two levels user location mechanism, where the SIP user address (called "Address of Record" or AoR) is mapped by the Registrar server to a new user URL (Uniform Resource Locator) referring to the SBC and then the SBC maps this new URL to the actual address of the user's UA. The presence of a backend Registrar server can let the overall architecture be more flexible and compatible with other server-based services, leaving the complete control of such services (e.g. AoR resolution or CPL (Call Processing Language)-based services) to the backend Registrar/Proxy. Following this approach, in our proposal the mobility of the terminal amongst different access networks is controlled by the MMS/SBC, while the external SIP Registrar simply points to the MMS/SBC of each registered user (this means that the user's AoR is simply mapped to a user-specific URL pointing to the MMS/SBC). The SBC/MMS will take care of NAT traversal, so that the MT can be reached by SIP signaling and can send/receive media flows even beyond a NAT. As for SIP signaling, the MMC in the MT and the MMS in the SBC implement the SIP extension described in [49] which allows the MMC to receive SIP responses on the same port where it sent corresponding SIP requests. A "keep-in-touch" mechanism is needed to keep the pinhole in the NAT open. Various techniques can be used [33] such as dummy UDP packets (from the MMC to the MMS or vice-versa), mal-formed SIP messages, well-formed SIP messages. We use periodic SIP register messages from MMC to MMS. The "keep-in-touch" packets are sent every 30 seconds, so they use a very limited amount of resources.

4.3.1 Location Update Registration: initial and “off-call” mobility management

The Location Update Registration is the basic mobility procedure that allows a MT to notify the MMS about its “position” (or better its IP address) and select the currently preferred interface for sending/receiving SIP signaling and media flows. The sequence diagram of this procedure is shown in Figure 57.A. The MMC in the MT sends a Registration Request to the MMS over the “selected” interface. When the 200 OK is received, the “keep-in-touch” mechanism is activated on that interface (and deactivated on the previous interface if needed). This procedure is activated at the start up of the MT (or when the MT first enters in a coverage area), or whenever the MT wants to change the selected interface if it is under coverage of more than one network. We can refer to this procedure as “off-call” mobility management because we assume that terminal is not engaged in a call. If the terminal is engaged in a call, the handover procedure will be executed (see later on).

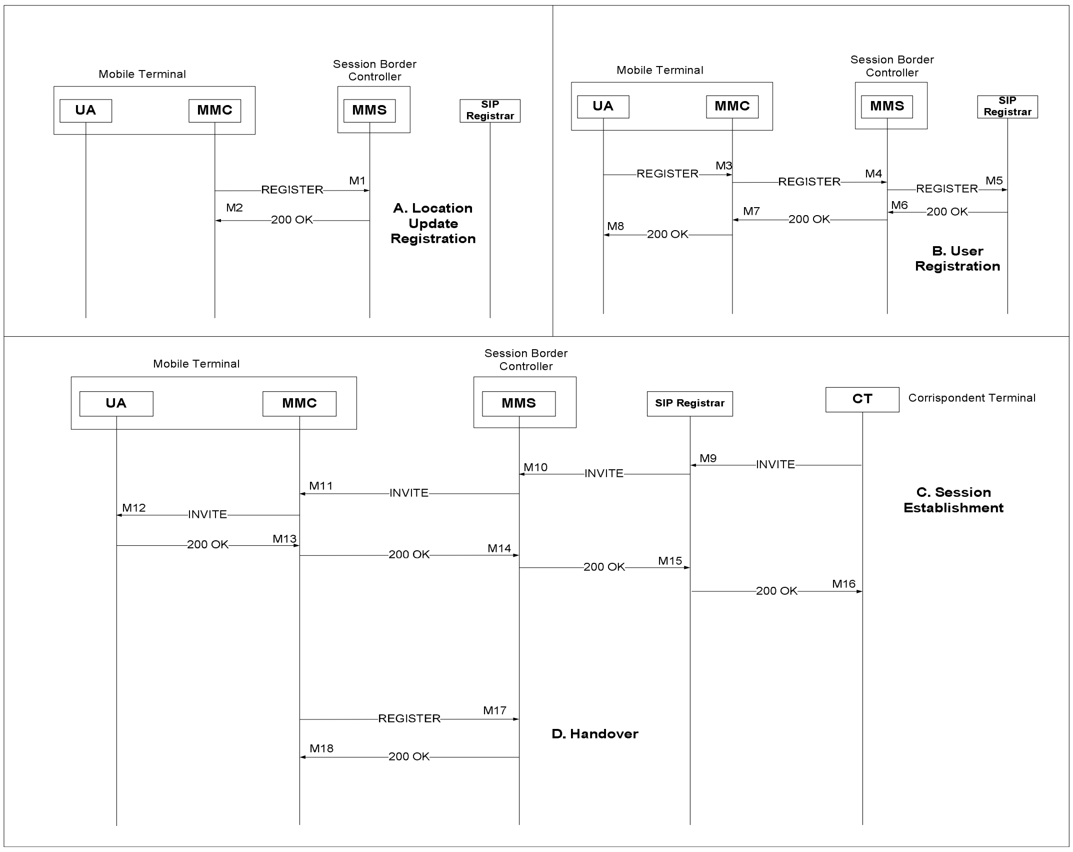

Figure 57 - Signaling procedures

As result of the Location Update Registration procedure, the SBC/MMS becomes aware of the current position of the MT, and can correctly route any new request or response messages addressed to the mobile UA. A key aspect concerning this procedure and its usage is the UA identification and addressing. In general a user may have more that one UA active, each one attached to a network with its own IP address (and SIP port). When sending a request or receiving a response, SIP usually identifies the users through the URLs present within the From and To header fields and through the request URI, while the actual address of the UA is normally present in the Via and Contact header fields. Unfortunately, neither the user URL nor the UA address can be used for UA identification since the former is not bound to a specific UA (more user's UAs can be present) while the latter changes each time the UA moves from one network to another and, in presence of NATs, it is not unique due to the normal reuse of private addresses. For this reason a proper UA identification mechanism would be needed, but current SIP standard does not provide such mechanism. We used an identifier that the MMC inserts in the Contact and in the Via header fields, and it is denoted as Terminal-ID in the SIP messages shown in Figure 58. The details of the solution have used in the testbed can be found in [50], which reports the complete SIP messages related to the various procedures.

4.3.2 User Registration

This procedure consists in the UA registration with its own SIP Registrar server (the backend SIP registrar). The sequence diagram of this procedure is described in Figure 57.B. As any other SIP message, when the UA sends its own registration request to the SIP Registrar, the message is sent by the UA to the MMC which is seen as outbound proxy. The MMC forwards it to the MMS. Acting on behalf of the MT, the MMS will forward the registration to the SIP Registrar, which will update the contact address associated with the user's AoR (that is the public user identifier). When forwarding the Register message, the MMS/SBC modifies the Contact header in such a way it becomes the new "contact" for the user. This is required in order to force the routing through the SBC/MMS of all further requests addressed to the user. Such mangling of the contact URL should be unique and reversible. It can be done in several ways, using either a stateless approach (e.g. by mapping the previous URL, opportunely stuffed, within the new URL) or a stateful one (e.g. by using a local mapping table). We have chosen a stateless approach. In message M5 of Figure 58 there is an example of the rewritten contact (further details can be found in [50]). From now on, only the MMS will keep track of the MT movements, while the SIP Registrar will just believe that the MT location is the IP address of the SBC.

4.3.3 Session Establishment

The session establishment procedure consists in a standard SIP session setup procedure. All session establishment messages for MT are handled by the SBC. Before relaying an INVITE request sent by the caller and the corresponding 200 OK response sent by the callee the SBC modifies the corresponding SDP bodies in order to act as RTP proxy for media flows in both directions. This is needed to correctly handle NAT traversal in the path towards the MT, and it is done by exploiting the symmetric RTP approach as in a typical SBC implementation.

Once the session is established, the media packets start to flow over the selected wireless interface. In principle, there is no need to send anything on the unselected active interfaces, that should be used only when an "on-call" mobility procedure occurs. On the other hand our practical experience suggested that starting sending the packets on the 3G interface introduces an initial delay that can be quite large and can cause noticeable disruption in the voice communication during the handoff. Therefore we introduce a "keep-alive" mechanism between MMC and MMS during the call phase: the MMC sends dummy UDP packets to the MMS over the unselected wireless interfaces. The MMS will take care of discarding the received keep-alive packets so that they are not forwarded to the CT.

Message 1: MMC to MMS REGISTER sip:MMS_IP SIP/2.0 Via: SIP/2.0/UDP Terminal_ID; branch=z9h To: <User_ID> From: < User_ID >;tag=dba Call-ID: 7bb@002 CSeq: 1 REGISTER Contact: <sip: Terminal_ID>	**Message 2: MMS to MMC** SIP/2.0 200 OK Via: SIP/2.0/UDP Terminal_ID;branch=z9h; received=IP_NAT To: <User_ID> From: < User_ID >;tag=dba Call-ID: 7bb@002 CSeq: 1 REGISTER	**Message 3: UA to MMC** REGISTER sip:domain.net SIP/2.0 Via: SIP/2.0/UDP MMC_IP;rport;branch=z9h From: <sip:user@domain.net>;tag=25 To: <sip:user@domain.net> Contact: <sip:User_ID@MMC_IP> Call-ID: FDA@domain CSeq:1 REGISTER
Message 4: MMC to MMS REGISTER sip:domain.net SIP/2.0 Via: SIP/2.0/UDP Terminal_ID;branch=z9h; Via: SIP/2.0/UDP MMC_IP;rport;branch=z9h From: <sip:user@domain.net>;tag=25 To: <sip:user@domain.net> Contact: <sip: Terminal_ID> Call-ID: FDA@domain CSeq: 1 REGISTER	**Message 5: MMS to Registrar** REGISTER sip:Reg_IP SIP/2.0 Via: SIP/2.0/UDP MMS_IP;branch=z9h Via: SIP/2.0/UDP Terminal_ID;branch=z9h; Via: SIP/2.0/UDP MMC_IP;rport;branch=z9h Route: <sip: Reg_IP;lr> From:<sip:User_ID@REG_IP>;tag=157 To:<sip:User_ID@REG_IP> Call-ID: FDA@domain CSeq: 1 REGISTER Contact: <sip:/MMS_ID- Terminal_ID@MMS_IP>	**Message 6: Registrar to MMS** SIP/2.0 200 OK Via: SIP/2.0/UDP MMS_IP;branch=z9h Via: SIP/2.0/UDP Terminal_ID;branch=z9h; Via: SIP/2.0/UDP MMC_IP;rport;branch=z9h From:<sip:User_ID@REG_IP>;tag=157 To:<sip:User_ID@REG_IP> Call-ID: FDA@domain CSeq: 1 REGISTER
Message 7: MMS to MMC SIP/2.0 200 OK Via: SIP/2.0/UDP Terminal_ID;branch=z9h; Via: SIP/2.0/UDP MMC_IP;rport;branch=z9h From: <sip:user@domain.net>;tag= 25 To: <sip:user@domain.net> Call-ID: FDA@domain CSeq: 1 REGISTER	**Message 8: MMC to UA** SIP/2.0 200 OK Via: SIP/2.0/UDP MMC_IP;rport;branch=z9h From: <sip:user@domain.net>;tag=25 To: <sip:user@domain.net> FDA@domain CSeq:1 REGISTER	**Message 9: CT to Proxy** INVITE sip:/MT_Terminal_ID@MMS_IP SIP/2.0 Via: SIP/2.0/UDP CT_IP; branch=z9h From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: <sip:CT_contact> Content-Length: 214
Message 10: Proxy to MMS INVITE sip:/ MMS_ID- MT_Terminal_ID@MMS_IP SIP/2.0 Via: SIP/2.0/UDP Proxy_IP;branch=z9h Via: SIP/2.0/UDP CT_IP From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: <sip:CT_contact> Content-Length: 214	**Message 11: MMS to MMC** INVITE sip:/ MT_Terminal_ID@ SIP/2.0 Via: SIP/2.0/UDP MMS_IP;branch=z9h Via: SIP/2.0/UDP Proxy_IP;branch=z9h Via: SIP/2.0/UDP CT_IP From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Record-Route: <sip:MMS_IP;lr> Call-ID: F16@192 CSeq: 1 INVITE Contact: <sip:/MMS_ID-/~CT_ID/AT- MMS_IP/PORT-5070@MMS_IP> Content-Length: 214	**Message 12: MMC to UA** INVITE sip:/ MT_Terminal_ID SIP/2.0 Via:SIP/2.0/UDP MT_Terminal_ID;branch=z9h Via: SIP/2.0/UDP MMS_IP;branch=z9h Via: SIP/2.0/UDP Proxy_IP;branch=z9h Via: SIP/2.0/UDP CT_IP From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: <sip:/MMS_ID-/~CT_ID/AT- MMS_IP/PORT-5070@MMS_IP> Content-Length: 214
Message 13: UA to MMC SIP/2.0 200 OK Via: SIP/2.0/UDP MT_Terminal_ID;branch=z9h Via: SIP/2.0/UDP MMS_IP;branch=z9h Via: SIP/2.0/UDP Proxy_IP;branch=z9h Via: SIP/2.0/UDP CT_IP From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: < sip:MT_User_ID@MMC_IP >	**Message 14: MMC to MMS** SIP/2.0 200 OK Via: SIP/2.0/UDP MMS_IP;branch=z9h Via: SIP/2.0/UDP Proxy_IP;branch=z9h Via: SIP/2.0/UDP CT_IP From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: < sip:MT_User_ID@MMC_IP >	**Message 15: MMS to Proxy** SIP/2.0 200 OK Via: SIP/2.0/UDP Proxy_IP;branch=z9h Via: SIP/2.0/UDP CT_IP From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: < sip:MT_User_ID@MMC_IP >
Message 16: Proxy to CT SIP/2.0 200 OK Via: SIP/2.0/UDP CT_IP; branch=z9h From: <sip:CTuser@domain.net>;tag=871 To: <sip:MTuser@domain.net> Call-ID: F16@192 CSeq: 1 INVITE Contact: < sip:MT_User_ID@MMC_IP >	**Message 17: MMC to MMS** REGISTER sip:MMS_IP SIP/2.0 Via: SIP/2.0/UDP Terminal_ID; branch=z9h To: <User_ID> From: < User_ID >;tag=dba Call-ID: 7bb@002 CSeq: 1 REGISTER Contact: <sip: Terminal_ID> HO_Call_Id:F16@192	**Message 18: MMS to MMC** SIP/2.0 200 OK Via: SIP/2.0/UDP Terminal_ID;branch=z9h; received=IP_NAT To: <User_ID> From: < User_ID >;tag=dba Call-ID: 7bb@002 CSeq: 1 REGISTER

Figure 58 - SIP messages

4.3.4 On-Call Mobility: the handover procedure

The on-call mobility management procedure takes place when the UA identifies the need for handoff during an ongoing VoIP session. In our proposal, all the handover signaling messages can be exchanged on the target network (this approach is commonly referred to as “forward” handover). Therefore the handover can be performed even if the communication on the old network is interrupted abruptly. The handover procedure is MT initiated. The MMC in the terminal sends an “handover” Register message over the target network interface addressed to the MMS in the SBC. Differently from a Location Update Register request, the handover Register request contains in the message body the reference to the active session to which the handover is referred.

At the same time, the MT starts duplicating the outgoing media packets on both interfaces (unless the old interface has gone down). As soon as the MMS in the SBC receives the Register message, it will start accepting packets coming from the new interface and discarding the ones coming from the old interface for the media flows corresponding to the call ID contained in a specific header in the handover Register message. Then it will send back the SIP 200 OK message to the MMC and start sending the media packets directed to the MT using the new interface. Thanks to the fact that the terminal has already started sending the packets on the new interface, the duration of the handover is minimized.

The most critical issue is that the “handover” Register message could be lost for any reason, delaying the handoff procedure. The standard SIP procedure foresees that the client performs a set of retransmission of the Register if the 200 OK is not received back. The SIP standard suggests a default value of the retransmission timeout equal to 500 ms, that is doubled on each retransmission. However this is not compatible with a reasonable performance of the handover in case of the loss of the Register message. Therefore we mandate that for the “handover” Register message a different duration of the retransmission timer is used. Register messages are sent with a fixed interval of 200 ms until the 200 OK is received or a transaction timeout occurs. The transaction timeout is set to 3 s corresponding to a maximum of 15 retransmissions. On the terminal side, the MMC will stop duplicating the packets on both interfaces as soon as the 200 OK is received or the first media packet is received on the new interface. Note that if the media packet is received, but no 200

OK message, the MMC will still continue sending the Register message until the Register transaction expires.

The criteria for taking the handover decision can be based on:

- the quality of the received signal (power, S/N ratio)
- on IP level measurements of the QoS in the path between the MT and the SBC over the different wireless networks (packet loss, jitter)
- on the cost of the connections.

4.3.5 Comparison with canonical SIP based mobility

In classical SIP based mobility, when the MT moves to a new access network changing its IP address during a call, it re-invites the remote CT in order to re-establish a new media streams (the handover is handled by the remote CT). Then the MT has to register the new address to the SIP registrar. Instead, in the proposed solution the two functions are tied in just one registration procedure between the MT and the SBC, while the corresponding terminal is let completely unaware of the MT movement. This increases the handover performances, increases the compatibility with legacy remote terminals that might not handle correctly the re-invitation procedure, guarantees a better privacy (since the position and movement of the MT are hidden by the SBC).

4.4 Implementation and Measurements

We have implemented the proposed solution and realized a testbed across our University campus network (both over WiFi and Bluetooth), a Wifi network connected to an operator's network (Telecom Italia) via ADSL and two different 3G networks (Vodafone and TIM). The testbed layout is shown in figure Figure 59. The Mobile Terminals has been implemented using laptops with Windows XP SP-2 (this version of XP is only required for Bluetooth), the SBC and the SIP Registrar are implemented on a standard PC (both Windows XP and Linux can be used). MMC and MMS have been implemented in Java using (and modifying) the open source MjSip Java SIP stack [40]. As SIP User Agent we used the Xlite software client [51]. The laptop is equipped with an internal WiFi card, with a PCMCIA card for 3G access, and a BT dongle compatible with XP SP-2. As WiFi access network we used both an our own Access Point connected to the Campus Fixed Lan, and a WiFi network in our labs which is connected to Telecom Italia backbone. As 3G network we used the Vodafone network and the TIM network. As Bluetooth access network we used a Linux host with a Bluetooth dongle and the open source “BlueZ” Bluetooth stack. The Linux host is configured to bridge the Bluetooth PAN with the fixed Ethernet LAN, so that a client host connecting to the PAN simply gets an IP address valid on the fixed Ethernet LAN.

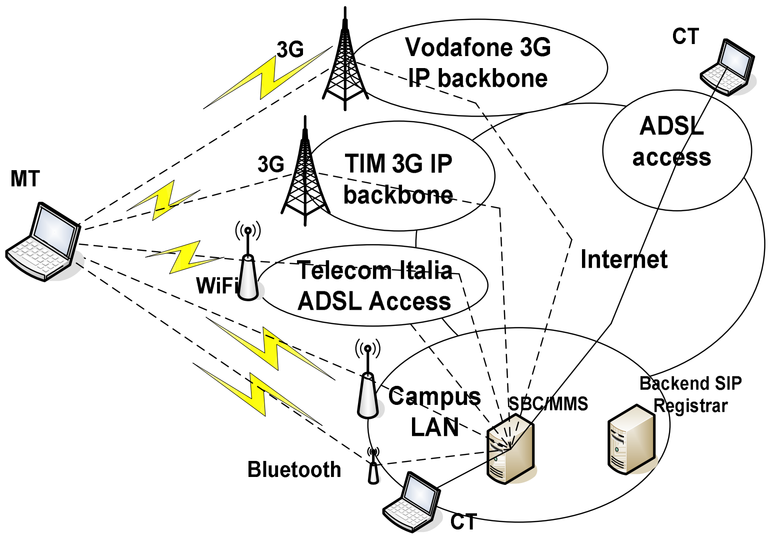

Figure 59 - Testbed layout

The SBC and SIP Registrar were located in our campus LAN and given a public IP address. As Correspondent Terminals we experimented both a PC in our campus LAN and a PC using an ADSL access.

In the Mobile Terminal, the MMC interacts with the operating system by checking the status of the interfaces with the “ipconfig” command. The MMC offers a simple Graphical User Interface which shows the currently active interfaces and allows to control the handover by choosing the “selected” interface.

No handover decision criteria are implemented in the described testbed. The handover decision is manually provided through the Graphical User Interface of the MMC.

On the testbed we first assessed the performance of the different access networks (in particular the 3G cellular networks) in the support of VoIP application (Section 4.4.1) and then the performance of the proposed handover procedure (section 4.4.2). We performed some subjective measurements of the perceived VoIP quality during the HO procedure and we found that the voice impairments are due to the different networks delays experimented by the WLAN (or Bluetooth) and the 3G networks (as shown later in Figure 61), while no impairment is perceived making the handover among two networks with the same delay. We are currently working on objective evaluation of voice quality using an approach [52] based upon a reduction of the ITU-T's E-Model [53] to transport level measurable quantities.

4.4.1 Evaluation of access network performance

In order to evaluate the performance of different access networks, we have developed a tool named “Throcalc” which is able to evaluate packet loss, Round Trip Time (RTT) and one-way delay jitter with powerful NAT traversal capability. The tool is composed of a client side which runs on a PC (both Windows and Linux OSs are supported) which can be equipped with any network interface and a server side which we run on a PC with public IP address on our university campus network (e.g. on the same host where the SBC/MMS is located). Therefore we are able to evaluate the performance of the “uplink” (from MT to SBC/MMS) and “downlink” (from SBC/MMS to MT) channels over the different wireless network. Note that the performance that we will consider is not only related to the wireless part of the path. For example when evaluating the performance of a 3G network, we include the fixed part of the radio access network, the IP backbone of the 3G operator, the Internet path from the 3G operator up to our campus network and finally the path from the campus network border router up to the SBC/MMS. Anyway this is exactly the path that will be crossed by voice packets that cross the SBC/MMS.

	BlueTooth	WiFi	3G net 1	3G net 2
Average packet loss ratio	0,11%	0,06%	0,79%	0,24%
Maximum packet loss ratio	0,13%	0,13%	2,89%	0,29%

Table 7 - Packet loss ratio of the different network access

A more detailed report of the measurement campaign can be found in [50], we only present here the main results. We measured a very good (i.e. low) loss rate using all the different access network. Table 7 reports a sample of our loss measurements over the different access network. On the other hand the RTT was not good when using both the 3G networks that we have tested (we recall that the measurements are related to the whole path from MT to SBC, which does not only include portions of 3G network, although we believe that the loss and delays are mainly related to the 3G portion of the path).

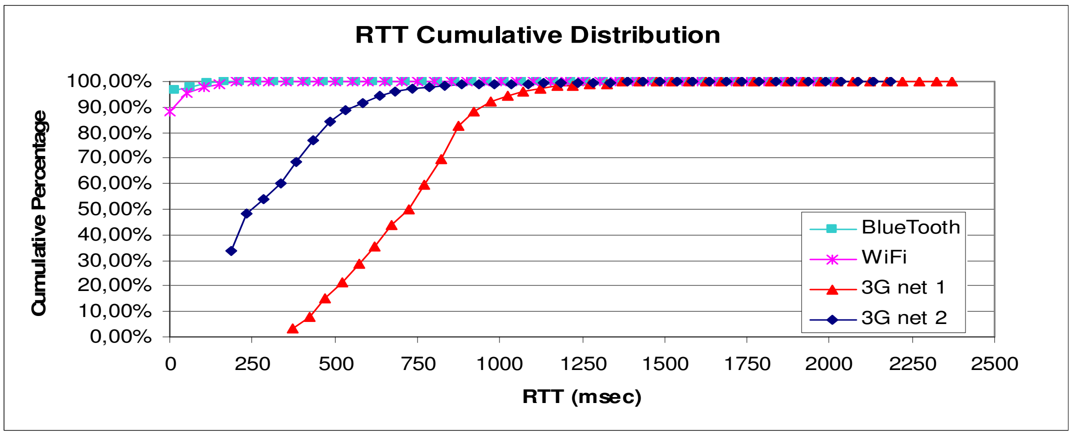

Figure 60 - Cumulative distribution of RTT for different access network

Figure 60 shows the cumulative distribution of RTT for the different access networks. Each distribution is evaluated with 5 different tests of duration 60 seconds repeated with 15 minutes interval during a working hour (e.g. 11 am). The average RTT is in the order of 400 ms and 800 ms for the two networks and even worse is the 95% percentile which is in the order of 600 and 1000ms, resulting in a degradation of voice experience.

4.4.2 Evaluation of handover performance

We analyzed the performance of the handover by capturing the media and signaling packets on the MT and on the SBC, using the Ethereal passive measurement tool [54]. We did not consider the path between the SBC and the Correspondent Terminal, as it does not impact the performance of the handover. The GSM codec at 13 kb/s was used. We have recorded the departure and arrival times of voice packets at the MT and at the SBC. We analyzed both the uplink flow (MT→SBC) and the downlink flow (SBC→MT) and we considered the handovers from WiFi to 3G and vice-versa (in total we have 4 scenarios).

Looking at the 4 graphs in Figure 61, in the x axis we put the departure time of packets from the originating interface, while in the y axis we put the arrival time of the packets at the destination interface. As the clocks are not synchronized, the time is relative to the first sent or received packet on the interface and we are not able to measure the absolute “one-way delay”. This is not a problem, as we are interested in the differential delay among arrived packets. For the different scenarios we will discuss: 1) the impact of the difference in the one-way delay between the WiFi and the 3G network during the handover; 2) the handover completion time, i.e. the time elapsed from when the MMC starts the handover procedure and when the procedure is completed and the voice in both directions is flowing on the target interface.

Let use define as U_{up} and U_{dn} the one way delay for the 3G network in the uplink (MT$\rightarrow$SBC) and downlink (SBC$\rightarrow$MT) direction. These delay do not only cover the 3G network, but all the path from MT to SBC, crossing the 3G network (see Figure 59). Similarly we define W_{up} and W_{dn} for the WiFi network. In the performed experiments, the measured round trip time between the MT and the SBC for the 3G access (i.e. $U_{up}+U_{dn}$) was in the range of 200 ms (as shown in Figure 62), while for the WiFi access (i.e. $W_{up}+W_{dn}$) was in the range of 20-25 ms.

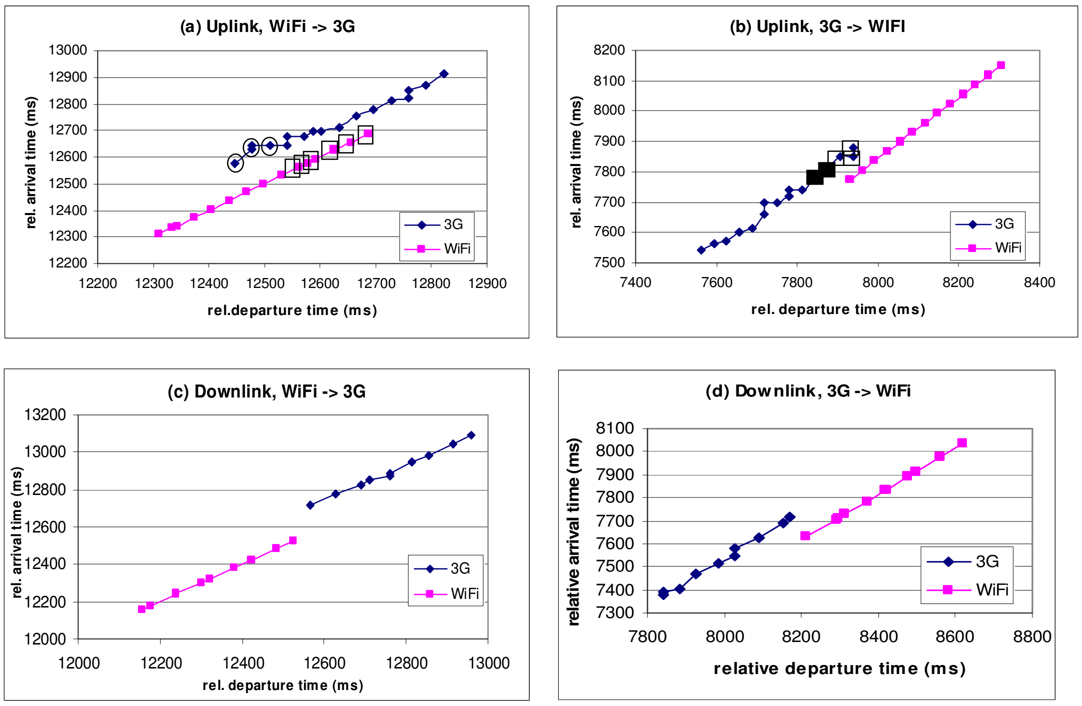

Figure 61 - RTP arrival patterns during handovers in 4 scenarios

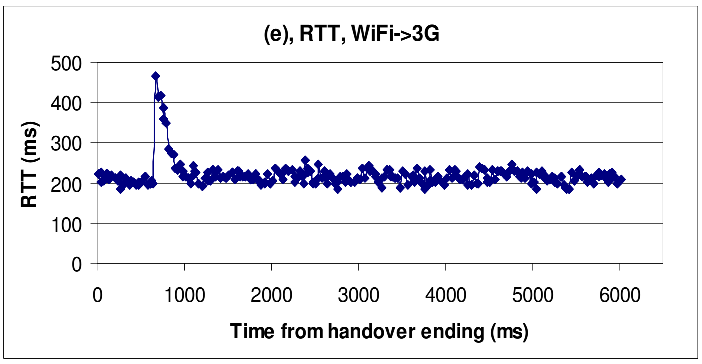

Figure 62 - RTT estimation after handover procedure ending (3G network)

Figure 61 reports the results for the 4 scenarios. From the diagrams related to uplink (a and b) we can give an estimate of the difference in the "one-way delay" for the 3G access and for the WiFi access in the "uplink" (i.e. U_{up}-W_{up}). As the packets are duplicated, the difference in the y axis between the arrival of the same packet sent on the WiFi and on the 3G interface is the delay difference. It turns out that at the time of our tests, uplink one-way delay experienced in the 3G access is 80 to 110 ms higher than the one experienced in WiFi. A set of packets will arrive from the 3G interface which are the copies of the already arrived packets. This packets, marked with a circle in Figure 61-a, will be forwarded and received by the CT as duplicated packets. The duration of this burst of duplicated packets is equal to the difference of the "uplink" one way delay between WiFi and 3G (U_{up}-W_{up}). As for the handover completion time, it roughly corresponds to the round trip time on the target interface (3G in this case: U_{up}+U_{dn}).

We measured 270 ms for the interval between the REGISTER and the 200 OK (i.e. the handover duration as perceived by the MT) in the test shown in Figure 61-a. As a confirmation, we can see that in Figure 61-a the packets are duplicated from t=12430 ms to t=12700 ms. This duration is almost entirely caused by the round trip time between MT and SBC/MMS. This is confirmed in Figure 62 which shows the RTT measured analyzing the traces of RTP packets captured on the MT and on the SBC/MMS for 6 seconds following the handover.

In case of the handover from 3G to WiFi (always in the uplink case), the MT sends the SIP REGISTER message on the "faster" WiFi network and starts duplicating the voice packets. When the SBC receives the REGISTER it will start accepting packets sent on the WiFi interface and discarding those sent on the 3G interface (marked with a square in Figure 61-b). The first received packets sent on the WiFi interface will have an higher sequence number than the last one received coming from the 3G interface, as the packets sent on the WiFi interface have "overcome" the ones sent on the 3G interface. A number of packets will be lost, and these packets are marked with the solid square in Figure 61-b. The duration of the burst of lost packets is again equal to the difference in the uplink one way delay between 3G and WiFi network. As for the handover completion time, we measured 32 ms for the interval between the REGISTER and the 200 OK in the test shown in Figure 61-b. In fact, the packets are duplicated from t=7906 ms to t=7938 ms. Coming to the downlink flows, let us consider the handover from WiFi to 3G

(Figure 61-c). The SBC will stop sending packets towards the WiFi network and start sending them towards the 3G network when the REGISTER message is received. The first packet sent towards the 3G network will experience an additional delay equal to the difference in the "downlink" one way delay between 3G and WiFi network (U_{dn}-W_{dn}) which is in the order of 200 ms in Figure 61-c. The gap shown does not represent a loss of some packets, it only shows a delay between the reception of the last packet sent on the WiFi interface and the reception of the first packet sent on the 3G interface.

Finally, let us consider the handover from 3G to WiFi for the downlink flow (Figure 61-d). As soon as the REGISTER message is received by the SBC the packets will be sent towards the WiFi interface and will arrive at the MT in advance with respect to packets with lower sequence number previously sent towards the 3G interface. The duration of the advance is equal to the difference in one-way delay (in the order of 200ms in Figure 61-d. Note that no packets are lost in this handover, the gap in Figure 61-d represents the timing advance of the first packets sent on the wifi interface that experience lower delay and arrive before the last packets sent on the 3G interface.

Similarly to what we have done in the uplink, from the data reported in Figure 61-c and d we can evaluate the difference in one-way delay for the downlink U_{dn}-W_{dn}. In our test we measured that 3G one-way downlink delay was from 80 to 110 ms higher than WiFi. The results show that the different delay between the WiFi and 3G network is a critical factor. If the differential delay is reasonably low the voice decoder is able to hide the handoff. In our tests, where U_{up}-W_{up} and U_{dn}-W_{dn} are in the order of 110 ms, the handovers are not perceived.

It is interesting to compare the results shown in Figure 61 with the corresponding measurements without using the keep-alive mechanism introduced in Section 4.3.3.

As we can see in Figure 63, which reports the uplink measurement for the WiFi to 3G handover, the initial differential delay between the 3G and WiFi is in the order of 2,8 s. Correspondingly, we have that the duration of the handover (during which all packets are duplicated) is in the order of 3 s. This is due to the fact that starting to transmit over a 3G interface requires a considerable amount of time. Just for comparison, we have reported the diagram of Figure 61-a in an arbitrary position in the left part of Figure 63.

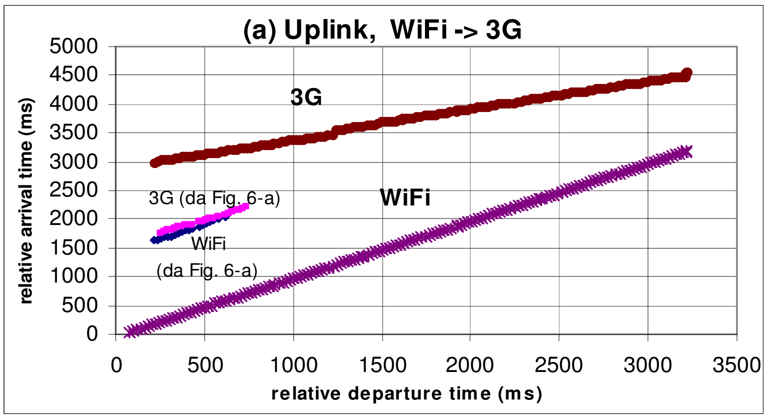

Figure 63 - RTP arrival pattern without keep-alive on the unselected 3G interface

It is possible to appreciate the difference in terms of handover duration and of the distance between 3G and WiFi packet arrival time. The conclusion is that the keep-alive mechanism is needed to support seamless handover. The results shown in Figure 61 consider the favorable case in which both interfaces remain active during the handover. It can happen that the old interface goes down suddenly and does not allow to transmit packets during the handover. In our solution, the uplink flows are not affected, as the MT starts sending packets on the new interface immediately. On the contrary, the downlink flows are affected, as the SBC will start transmitting packets towards the new interface only after receiving the handover request from the MT. We have analyzed this case with temporal diagrams similar to the one shown in Figure 64 and we have repeated the measurements of handover performance.

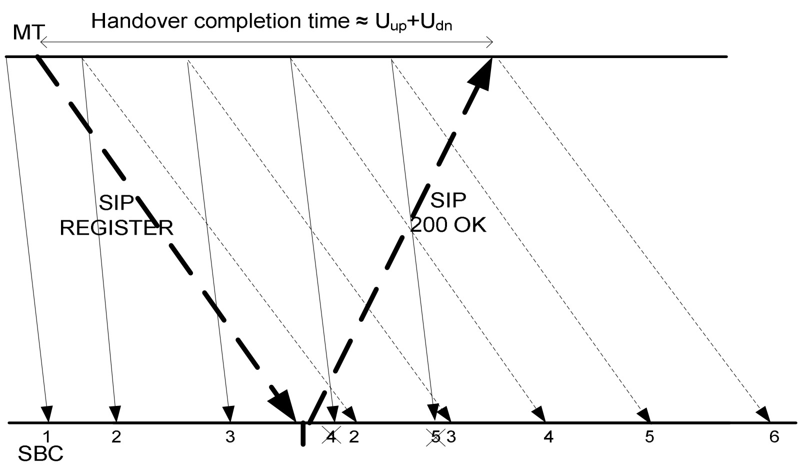

Figure 64 - Temporal diagram for WiFi → 3G handover

The full results are not shown here, anyway we have found theoretically and measured from the testbed that on the downlink flow we have an impairment in the order of $U_{up} + W_{dn}$ for the handover 3G→WiFi and in the order of $W_{up} + U_{up}$ for the handover WiFi→3G.

Similarly to what we have done for WiFi and 3G networks, we evaluated the performance of our handover mechanism, switching from Bluetooth to 3G network and vice versa. The performances of the BT network are very similar to the behavior of WiFi network as shown in Table 7 and in Figure 60. To support this statement, in [50] we report all the uplink and downlink measurement results for the handover between Bluetooth and 3G. In one sample measurement reported, we obtained 265 ms for the interval between the REGISTER and the 200 OK (i.e. the handover duration as perceived by the MT).

The reported results show that the impairments in the handover procedure are due to the intrinsic RTT of the “target” network and to the differential one way-delays between the origin and the target network.

4.5 Related Work

A survey on the different mobility management approaches to support mobility in VoIP services can be found in [55]. Mobility mechanisms for IP networks can be classified in IP network layer, transport layer or on application layer mechanisms. Mobility is provided at IP layer by the Mobile IP (MIP) [56]. Mobile IP is not directly related to VoIP applications, since it is completely transparent to the upper layers performing the same behavior for all incoming/outgoing IP datagrams. It provides the MT with a single IP address that “follows” the terminal in its wandering. The main disadvantage of MIP approach is that it needs a support from the routers in each access network. A MIP terminal will be able to roam only over MIP enabled access network.

SIP-based solutions belong to application layer mechanisms and they are usually based on SIP signaling for re-negotiating the media sessions when a handover occurs, as described also in [41]. In the classical SIP based approach the handover is performed end-to-end, only involving the MT and the CT.

In both the approaches (MIP and SIP) there is the need to minimize the service disruption during a handoff procedure and this issue has been addressed in several works (see for example [57] and [45]). An interesting work on SIP-based mobility can be found in [46]. The approach proposed in [46] is similar to the one exploited by our proposal, with some fundamental differences. In [46] the authors propose to re-negotiate the session with the remote UA (assisted by an intermediate SIP node) while we propose to use only the intermediate node (the SBC/MMS), with the main advantages of: i) reducing the latency of the overall handover procedure ii) overcoming eventual compatibility problems introduced by legacy remote UAs – in fact, in our proposal the remote UA is completely unaware of the mobility procedure. Note that the latter aspect has also some useful security implication (i.e. the remote UA can not trace the complete movement and the current position of the MT). Another important difference is related to the mode of operation of the intermediate node that in our solution is proxy-like instead of the B2BUA model proposed in [46]. We think that a proxy-style behavior is more flexible, less processor consuming, introduces less latency, and reduces the possibility of signaling incompatibility.

Other interesting works address the issue of 3G/WLAN interworking/integration, both in the research community (see for example [58] and [59]) and in standardization for a like 3GPP or 3GPP2. The underlying idea is to include WLAN access in the set of services provided by 3G operators to their subscribers. Most of the work done focused on the authentication and security aspects, while the issues related to handoff are still to be investigated.

In our work we assume that the terminal is able to authenticate (separately or in an integrated manner) to the different access networks and we focus on the problem of seamless mobility amongst heterogeneous networks and vertical handoff management. Similar focus can be found in [60], which considers the classical SIP based mobility (the handoff is handled by the correspondent terminal) rather than our solution to enhance the SBC.

4.6 Conclusions

In this Chapter we have presented a solution for seamless vertical handover between heterogeneous networks like WiFi and 3G, based on SIP. The novelty of the solution is that it is strongly coupled with the NAT traversal features provided by the so called Session Border Controllers. Assuming that the Mobile Terminals will be mainly roaming on networks with private IP numbering, the mediation of an SBC is already proposed to solve the NAT traversal issue, therefore we straightforwardly propose to enhance SBC functionality to support the mobility. The proposed solution can be exploited in the short term by a 3G operator willing to extend its services to WLAN, by a VoIP provider that uses the 3G network as IP transport and by an enterprise that wants to directly manage its voice services. In the long term this kind of approach will likely need to be included in 4G networks, which aim to support communication over heterogeneous networks (including legacy networks) in a seamless way.

All the proposed mechanisms have been implemented within a testbed that fully demonstrates the correctness and simplicity of the solution. Moreover, significant measurement tests have also be run in order to provide quantitative evaluation of the roaming solution.

Ongoing work concerns the realization of the mechanisms to drive the handover decision (both collecting the signal quality information from the network interface cards and making IP level measurements during the call active phase).

5 Link Layer: Multiple-Path Routing and Load Balancing Approach for WIMN

5.1 Introduction

This Chapter presents a preliminary study of a general layer-2 approach for routing and load balancing in Wireless Infrastructure Mesh Network.

The key idea is dynamic select routes among a set of slowly changing alternative network paths. Our approach decouples the routing and load balancing problem into two distinct sub problems: path creation and path selection. Paths are created through the reuse of classical 802.1Q multiple spanning tree mechanisms. This guarantees that, for each formed tree, a path is deployed from each mesh node to the Mesh Gateway. Moreover, each tree (path) is assigned a Virtual LAN identifier. Path selection is driven by a local algorithm running at each mesh node, fed by measurements (taken along each path connecting the mesh node to the gateway) which allow to dynamically determine which are the best paths. In order to route a packet it is sufficient to mark the packet with the VLAN tag corresponding to the chosen path. The described approach provides a very general and flexible framework: performance/stability trade-offs can be tuned through the choice of i) the mechanism used to measure the path quality; ii) the algorithm employed to select the path, and iii) the system parameter used (link costs and link weights) for the multiple spanning tree formation.

5.2 Motivations

The emergence and growth of many companies (such as Tropos Network, BelAir, FireTide, MeshDynamics, etc.) specialized in the provisioning of wireless infrastructure solutions, as well as the recent launch of standardization activities (such as 802.11s), demonstrate that Wireless Infrastructure Mesh Networks (WIMN) may represent a viable and cost-effective alternative to traditional wired infrastructure access networks. Unlike traditional routing algorithms [61] [62] [63] [64], designed for general ad hoc networks, our proposal is specifically devised to benefit from the unique characteristics of WIMN, i.e.: i) static mesh nodes, and ii) traffic mostly addressed from/to backhaul gateways. Furthermore, another motivation underlying our proposal is to provide a layer-2 approach which attempts to reuse as much as possible well established and commercially available 802.1D/Q bridging and access control techniques. This provides a number of assets, including the ease of integration of wired infrastructure segments into a wireless Mesh network infrastructure deployment, and the ability to see the whole network as a single layer-2 802.11 Extended Service Set, thus inheriting the management approaches therein devised (e.g. 802.11f IAPP or the currently under standardization IETF CAP-WAP protocol). Moreover, a layer-2 approach is in line with the requirements recently set in the 802.11s Task Group.

5.3 Approach

Layer-2 Ethernet Switched networks rely on spanning tree as forwarding/routing mechanism. However, its adoption is WIMN is questionable. Even if the Mesh topology is static, we expect that link qualities may nevertheless vary in time, especially if they furthermore depend on the relevant traffic load. Indeed, extensive literature paper shows that the routing effectiveness highly benefits from the adoption of dynamic metrics capable to account for both channel quality and load (see e.g. [65] [66]). However, by relying on time-varying link costs, frequent rearrangements of the spanning tree would be required. This is a costly process that causes abrupt interruption of connectivity. Even if the Rapid Spanning Tree protocol is employed, unacceptable impairments are deemed to occur.

Our proposal relies on spanning tree, but in a quite different manner, and consists in i) creating quasi static paths through a spanning tree protocol, and ii) dynamically route and balance the load through appropriate selection of the deployed paths. The next section illustrate into additional details these two sub-problems.

5.3.1 Multiple Path creation

Rather than using a single time-varying tree, we deploy partially overlapping spanning trees, meant to remain quasi static (i.e. changes occur only upon significant events such as link failures). This is accomplished by slightly adapting, to the WIMN context, the Multiple Spanning Tree Protocol (MSTP) specified in the 802.1Q standard, and, most important, by using the related Virtual LAN (VLAN) identifiers therein assigned and managed, in a significantly different manner. In addition, our solution provides the key advantage that more than 1 backhaul gateway can be managed.

Specifically, let N be the number of gateways. We configure the network as the superposition of $M \geq N$ VLANs: for convenience of presentation, assume (not restrictively) that $M = K \cdot N$, where K is the number of alternative paths to be deployed per each gateway. As well known from 802.1Q, the MSTP protocol deploys one single spanning tree per each VLAN. We configure the MSTP so that each gateway is the root for K trees. Furthermore, through appropriate per-bridge (i.e. Mesh router) configuration it is possible[20] to differentiate the trees converging to a same gateway in order to avoid that they collapse into a single one tree. The above described operation deploys M static trees covering the whole network. Trees are managed by the MSTP operation in a fully standard manner, thus reconfigured only upon link failures. We underline that this provides each Mesh node with M alternative paths towards the backhaul gateways. Frames may be forwarded along any of this path by simply tagging them with the VLAN identifier corresponding to the selected path (the standard 802.1Q bridging operation guarantees that a VLAN tagged packet is routed accordingly).

5.3.2 Dynamic Path Selection

First, notice that a Mesh router exerts a very different treatment to frames, depending on the fact that they are locally originated (i.e. incoming from the clients associated to the node), or forwarded (i.e. received by an adjacent Mesh Router). In the latter case, frames are VLAN tagged, and their forwarding decision is hence fully determined by the 802.1Q forwarding operation (in turns of course depending on the relevant VLAN tagging). Conversely, locally originated frames are untagged (the end mesh clients are not involved in the MSTP operation). Dynamic path selection is applied only to the local frames: the decision of which path to take is delegated to a locally running path selection mechanism which receives as input the quality of the avail- able paths, and takes the routing decision by tagging the locally received frame with the VLAN identifier of the path.

The approach proposed in this work does not depend on which specific path selection mechanism is used: in terms of network operation, this can be considered as a black box whose output is the choice of the frame VLAN tag. However, it is

[20] Specifically, through appropriate setting of the MSTI priority vectors associated to each Bridge.

obvious that the performance of the proposed approach dramatically depend on how this algorithm is designed, and specifically: i) according to which criteria path quality is evaluated, and ii) whether - and how - hysteresis are considered in the path selection procedure[21].

At this stage of this work, we have not yet faced the issue of optimizing the network operation, and we have simply relied (as a proof of concept) on gross heuristics.

5.3.2.1 Path quality measurements

Our approach requires to associate a cost to each path. Since several paths are deployed (assuming 100 nodes and 10 trees, as much as 1000 paths should be assigned an explicit costs!), the approach devised to quantify such cost should be scalable and least invasive as possible. This forces us to exclude per-path active approaches where costs are computed through a per-path probing. A more natural approach is therefore to compute the cost associated to a single link, and combine such costs for determining the path cost. It is convenient to combine link costs through an additive process. This allows to incrementally compute the path cost along the tree in a manner identical to what the standard spanning tree protocol BPDU does. To this purpose, we have introduced specially labelled BPDUs which simply use, instead of the fixed link cost used for the spanning tree formation (in our case a same constant value for all the active network links), a dynamically updated link cost. These BPDUs are periodically generated and forwarded as standard STP-BPDUs, as they only differ in their cost field contents and not in their format. Computation and maintenance of such a per-link dynamic cost is delegated to each mesh node. Any proposed link cost metric for which additivity is reasonable (such as the widely adopted Expected Transmission Time [67]) can be used. How to optimize such a metric for the application to our considered framework is a current research issue.

5.3.2.2 Path selection decision

Our current selection algorithm is based on gross preliminary heuristics, which nevertheless have allowed us to understand some general requirements of the path

[21] It is intuitive that a load balancing mechanism devised to send traffic over the less instantaneously loaded path would suffer, in the best cases, of flapping phenomena, and, in the worst case, of dramatic instability.

selection mechanism. First, we noticed that it is important to de-synchronize the mesh node operation, as this would raise severe traffic oscillations. To this purpose, mesh nodes do not base their per-frame tag decision on the bases of the instantaneous path cost (i.e. that carried in the most recently received BPDU). rather, each node randomly sets a timer, samples the path cost load, and maintain the tag decision until the next timer expires. In addition, the decision whether to change path includes hysteresis thresholds, so that a path is changed only if the difference in cost with respect to the actual chosen path is greater than a given amount. Also in this case, we have found that using the same threshold in all the mesh nodes can be critical. Therefore, hysteresis threshold are also randomly modified when the previously mentioned timer expires. Whether these two mechanisms should be used in conjunction, or one of the two is sufficient (as we believe, if properly tuned settings are provided) is object of current research issue.

5.4 Conclusions

In this Chapter, although with very early algorithmic ideas, a preliminary study of a generic layer-2 framework for routing and load balancing in WIMN has been presented.

Deeper studies are in progress in order to: i)choose a dynamic link cost metric, ii)improve the path selection phase, migrating from heuristic settings to theoretically guided ones, iii) implement an NS-2 simulation platform for multi-rate scenarios and for a better support of the physical level.

Finally, compare the performance of the proposed framework with other routing approaches is necessary.

References

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

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

[3] S. Pilosof, R. Ramjee, Y. Shavitt, P. Sinha, “Understanding TCP fairness over Wireless LAN”, IEEE INFOCOM 2003, San Francisco, USA, March 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”, IST Mobile & Wireless Communications Summit, Aveiro, Portugal, June 2003

[5] S. Shenker, J. Wroclawski, “Network Element Service Specification Template”, RFC 2216, September 1997

[6] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/

[7] 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

[8] W. Stevens, “TCP Congestion Control”, RFC 2001

[9] S. Floyd et al., “The NewReno Modification to TCP's Fast Recovery Algorithm”, RFC 2582

[10] M. Mathis et al., “TCP Selective Acknowledgement Options”, RFC 201

[11] Linux Advanced Routing & Traffic Control - LARTC, http://lartc.org/

[12] http://netgroup.uniroma2.it/download/wlan_arc.tar.gz

[13] KIM et al., “Distributed Access Time Control for Per-Station Fairness in Infrastructure WLANs”, IEICE Trans Commun.2006; E89-B: 2572-2579

[14] N. Blefari-Melazzi, A. Detti, A. Ordine, S. Salsano, "Controlling TCP Fairness in WLAN access networks using a Rate Limiter approach," IEEE 2nd International Symposium on Wireless Communication Systems, pp. 375- 379, Sept. 2005

[15] L. Xiaoyang, C. Xiaolin, J.K. Muppala, "VQ-RED: An efficient virtual queue management approach to improve fairness in infrastructure WLAN," The IEEE Conference on Local Computer Networks, Vol. 7, 15-17 Nov. 2005

[16] X. Wang, F. Liang, "TFBR: A New Queue Management Guaranteeing TCP Fairness Based on TCP Advertised Receiving Windows Over WLAN," IEEE 2nd International Conference on Mobile Technology, Applications and Systems Vol. 1, 15-17 2005

[17] Yi Wu; Zhisheng Niu; Janfeng Zhu, "Upstream/downstream unfairness issue of TCP over wireless LANs with per-flow queueing," IEEE International Conference on Communications, vol.5, no.pp. 3543- 3547 Vol. 5, 16-20 May 2005

[18] J. Padhye, V. Firoiu, D.F. Towsley, J.F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," IEEE/ACM Transactions on Networking, Vol.8, 133-145, Apr 2000

[19] F. Liang, X. Wang, L. Xu, J. Fan, "On AP buffer effect upon TCP fairness over WLAN," IEEE International Conference on Mobile Technology, Applications and Systems, Vol. 4, 15-17 Nov. 2005

[20] R. Battiti, R. Lo Cigno, M. Sabel, F. Orava, and B. Pehrson, “Wireless LANs: from warchalking to open access networks.” Mobile Networks and Applications, Vol. 10, 275–287, 2005

[21] StockholmOpen Project. http://www.stockholmopen.net/

[22] B. Pehrson, K. Lundgren, and L. Ramfelt, “Open.Net – open operator neutral access network.” 12-th IEEE workshop on Local and Metropolitan Area Networks, Stockholm, SE, 2002

[23] E. Pelletta, F. Lilieblad, M. Hedenfalk, and B. Pehrson, “The design and implementation of an operator neutral open wireless access network at the kista it-university” 12-th IEEE Workshop on Local and Metropolitan Area Networks, 2002

[24] Wifidog, http://dev.wifidog.org/

[25] R. Flickenger, Building Wireless Community Networks. O’Reilly, 2003

[26] IEEE 802.1X: Standard for Local and metropolitan area network – Port-Based Network Access Control. IEEE, 2001 (Revision 2004)

[27] IEEE 802.11i: Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks. IEEE, 2004

[28] Shibboleth, http://shibboleth.internet2.edu/

[29] Athens, http://www.athensams.net/

[30] IRAP: International Roaming Access Protocol, http://www.irap.nl/

[31] Eduroam: Education Roaming, http://www.eduroam.org/

[32] M. Brunato, D. Severina, “WilmaGate: a New Open Access Gateway for Hotspot Management”, WMASH2005, pp. 56–64, 2005

[33] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks M. Handley and E. Schooler, “SIP: Session Initiation Protocol" IETF RFC 3261, June 2002

[34] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen and L. Stewart, “HTTP authentication: Basic and Digest Access Authentication”, IETF RFC 2617, June 1999.

[35] S. Salsano, G. Martinello and L. Veltri, "Wireless LAN-3G Integration: Unified Mechanisms for Secure Authentication based on SIP. IEEE international Conference on Communications" ICC 2006, Istanbul, June 2006

[36] 3GPP TS 33.102: "Security architecture"

[37] 3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)"

[38] TWELVE project homepage, http://dit.unitn.it/twelve/

[39] SIPfoundry reSIProcate: an rfc3261 sip stack, http://www.sipfoundry.org/reSIProcate/

[40] MjSIP – GPL open source SIP stack Java implementation, http://www.mjsip.org

[41] H. Schulzrinne and E. Wedlund, "Application-Layer Mobility Using SIP", Mobile Comp. and Commun. Rev., vol. 4, no. 3, July 2000

[42] J. Cumming "Sip Market Overview", http://www.dataconnection.com/network/download/whitepapers/sipoverview.pdf

[43] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", IETF RFC 3489, March 2003

[44] J. Rosenberg, J., "Obtaining Relay Addresses from Simple Traversal Underneath NAT (STUN)", draft-ietf-behave-turn-02 (work in progress), October 2006

[45] A. Dutta, S. Madhani, W. Chen, O. Altintas, H. Schulzrinne "Fast-handoff Schemes for Application Layer Mobility Management", PIMRC, Spain, 2004

[46] N. Banerjee, A. Acharya, S.K. Das, "Seamless SIP-Based Mobility for Multimedia Applications", IEEE Network, March/April 2006

[47] MediaProxy, http://mediaproxy.ag-projects.com/README

[48] M. Handley and V. Jacobson, "SDP: Session Description Protocol," IETF RFC 2327, April 1998

[49] J. Rosenberg and H.Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", IETF RFC 3581, August 2003

[50] M.Fiorani, A. Labella, A. Ordine, A. Polidoro, S. Salsano, L. Veltri, "Report of architecture and measurements for SBC based vertical handovers", May 2006, available at: http://netgroup.uniroma2.it/Stefano_Salsano/SIP-SBC-seam-HO/report-2006-05.pdf

[51] Xlite SIP User Agent, http://www.counterpath.com/

[52] R.G. Cole, J.H. Rosenbluth, "Voice over IP Performance monitoring", ACM SIGCOMM Computer Communication Review, Vol. 31 , Issue 2, pp. 9–24, April 2001

[53] ITU-T Recommendation G.107, "The E-Model, a computational model for use in transmission planning", December 1998

[54] G. Combs *et al.*, "Ethereal: A Network Protocol Analyzer", http://www.ethereal.com/

[55] N. Banerjee, Wei Wu; S.K. Das, "Mobility support in Wireless Internet", IEEE Wireless Communications, Vol 10, no 5, pp54-61, 2003

[56] C. Perkins et al. "Mobility support ofr IPv4" RFC 3344

[57] A. Campbell et al., "Design, implementation and evaluation of cellular IP", IEEE Personal Communications Magazine, vol. 7, pp. 42-49, Aug 2000

[58] M. M. Buddhikot, G. Chandranmenon, S. Han, Y.-W. Lee, S. Miller, L. Salgarelli, "Design and Implementation of a WLAN/CDMA 2000 Interworking Architecture", IEEE Communication Magazine, November 2003

[59] G. Ruggeri, A. Iera, S. Polito: "802.11-Based Wireless-LAN and UMTS interworking: requirements, proposed solutions and open issue". Computer Networks 47(2): 151-166, 2005

[60] A. Dutta, B. Kim, T. Zhang, S. Baba, K. Taniuchi, Y. Ohba, H. Schulzrinne "Experimental Analysis of Multi Interface Mobility Management with SIP and MIP", IEEE Wirelesscom, Maui, Hawaii, USA, June 2005

[61] C.E. Perkins and P. Bhagwat: "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers". Communications architectures, protocols and applications, pages 234–244, 1994

[62] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot: "Optimized Link State Routing Protocol". IEEE INMIC, Lahore, Pakistan, December 2001

[63] C.E. Perkins et al: "Ad hoc networking". Addison-Wesley Boston, chapter: Ad hoc On-Demand Distance Vector Routing, 2001

[64] D.B. Johnson, D.A. Maltz, J. Broch. DSR: "The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks". Ad Hoc Networking, pages 139–172, 2001

[65] R. Draves, J. Padhye, and B. Zill: "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks". SIGCOMM, Portland, Oregon, USA, Semptember 2004

[66] B. Awerbuch, D. Holmer, H. Rubens: "High throughput route selection in multi-rate ad hoc wireless networks", First Working Conference on Wireless On-demand Network Systems, WONS, Madonna di Campiglio, Italy, January 2004

[67] R. Draves, J. Padhye, B. Zill: "Routing in Multi-radio, Multi-hop Wireless Mesh Networks". ACM MobiCom, Philadelphia, Pennsylvania, USA, October 2004

[68] C. Burmeister, U. Killat, J. Bachmann, "TCP over Rate-Adaptive WLAN - An Analytical Model and its Simulative Verification," in Proc. IEEE WOWMOM 2006, pp. 339-348

[69] 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," IEEE Transactions on Wireless Communications, vol.6, no.4, pp. 1346-1355

[70] J. Semke, J. Mahdavi, and M. Mathis, “Automatic TCP buffer tuning,” in Proc. ACMSIGCOMM Symp. Communications Architectures Protocols, Vancouver, BC, Canada, Sept. 1998, pp. 315–323

[71] Joseph Davies, ”The Cable Guy: TCP Receive Window Auto-Tuning,” Microsoft TechNet Magazine, January 2007, available at http://www.microsoft.com/technet/technetmag/issues/2007/01/

[72] G. Bianchi, A. Di Stefano, C. Giaconia, L. Scalia, G. Terrazzino, I. Tinnirello, “Experimental Assessment of the Backoff Behavior of Commercial IEEE 802.11b Network Cards,” in Proc. IEEE INFOCOM 2007, pp. 1181-1189

[73] R. Braden, "Requirements for Internet Hosts - Communication Layers," IETF RFC 1122, October 1989

[74] Forsythe, G. E., M. A. Malcolm, and C. B. Moler, “Computer Methods for Mathematical Computations,” Prentice-Hall, 1976

[75] N. Blefari-Melazzi, A. Detti, A. Ordine, S. Salsano, "A mechanism to enforce TCP Fairness in 802.11 wireless LANs and its performance evaluation in a real test-bed," IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks, 2007 (WoWMoM 2007),pp.1-7, June 2007

[76] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, "Performance anomaly of 802.11b," IEEE Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2003, March 2003, vol.2, pp. 836-84

Appendix I ANALYSIS WITH DIFFERENT VERSIONS OF TCP

In order to assess the impact of different TCP version, we have repeated the simulation analysis described in Section 1.2.1 by using TCP NewReno and SACK.

The results confirm that both the upstream/downstream critical unfairness and the critical unfairness among upstream connections are not dependent on the TCP version. We report the simulation results only for the critical unfairness between upstream connections. Figure 65 reports the throughput of each upstream station, while varying both the number of stations (N) and the TCP version.

We can see that the occurrence of the starvation phenomena is not related with the TCP version. We note that the number of upstream connections that are able to start (i.e., with a throughput greater than zero) is similar for each TCP version taken into account. For instance, let us focus our attention on the N=10 case; the graph shows that only seven upstream connections among ten achieve throughput values greater than zero; that is, three connections are starving.

This independence of the TCP version is quite to be expected, being the starvation a problem tied with the TCP startup phase, during which the different TCP versions behave in a similar way.

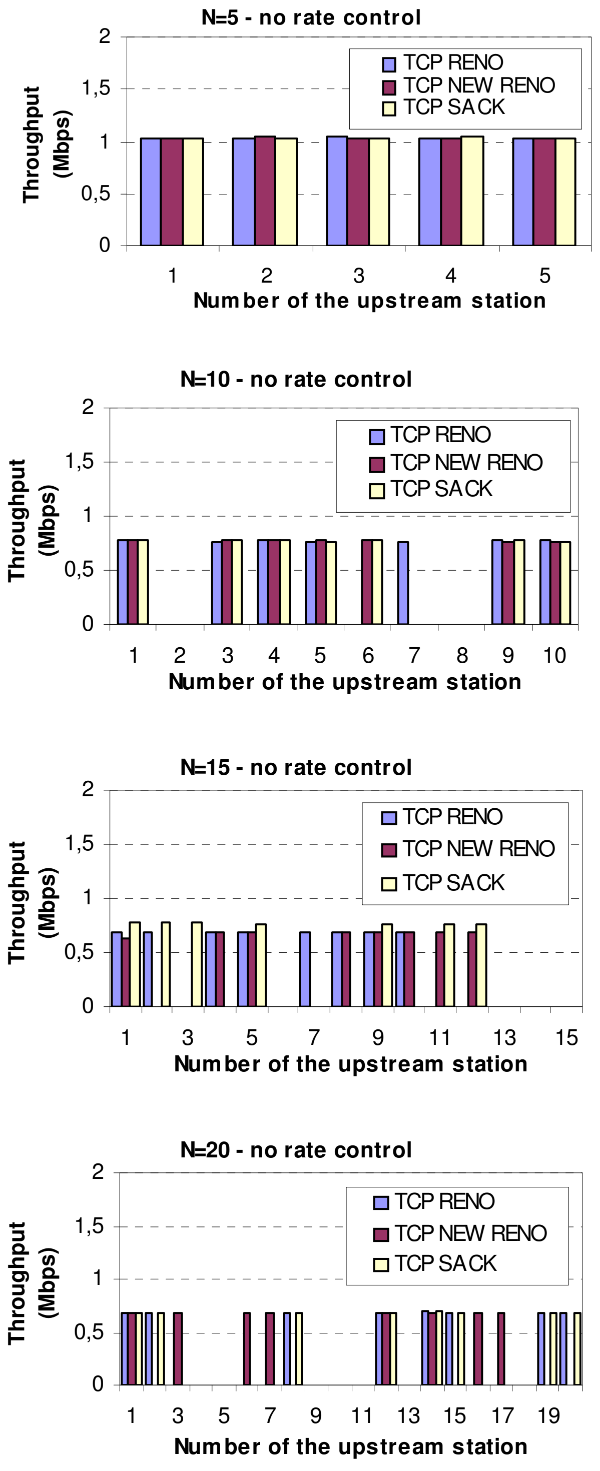

Figure 65 - Upstream TCP connection throughput in the "no rate control" scenario

Appendix II RATE ESTIMATION ALGORITHM

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 $\alpha = \min((T-T_{old})/\tau,1)$
```

Appendix III ASSUMPTIONS AND MODEL LIMITS

In this appendix we discuss the assumptions and the ensuing model limits (presented in Section 2.3).

Assumption a1 (Table 5) means that the wireless part is the bottleneck. Neglecting the packet loss in the wired part seems reasonable as loss phenomena mainly occur in the wireless part. On the contrary, the delay suffered in the wired part can not be always neglected. Our basic model is derived for those cases in which the delay assumption holds true; however in this work we also say how to modify the model to take into account the wired part delay.

Assumptions a2 is reasonable, since the operative system of a wireless host usually allocates a large amount of memory to its network interfaces (e.g. 1000 packets).

As regards assumption a3, the Reno version with delayed ACK is the most widely deployed on current operative systems. So this assumption is almost always valid.

Assumption a4 is the major model limit. With it, we assume that all the TCP connections start and reach a steady-state behavior. In Chapter 1 we show that this assumption is not always true. As a matter of fact, in case of heavy losses, some TCP connections may be completely starved. We are not able to capture this critical-starvation phenomenon; thus, our model is valid when the loss probability is such that critical starvation does not occur. When critical starvation does occur the model results can be considered as a best-case in term of fairness.

As regards assumption a6, at least in the long term[22] 802.11 MAC layer succeeds to provide a perfect per-packet sharing among STAs. Nevertheless, in [72] the authors show that different implementations of MAC layer may lead to a different medium access probability among STAs and AP. In this work we account of this fact through a parameter β_i, which represents the ratio of the AP and i-th STA medium access probabilities.

Assumption a5 saves us from evaluating the average value of the occupancy of the AP buffer, by assuming a full buffer occupancy, when losses can occur. In the no-loss case, this assumption is not necessary. The assumption is quite true as TCP

[22] In the short term, it may occurs that backoff procedures do not succeed in perfectly sharing the channel capacity on a per-packet basis (due to collisions and idle times). Anyway, the testbed result will show that this phenomenon is negligible.

flows tend to saturate the AP buffer, being this device the network bottleneck. The higher the number of connections the more verified a5 is[23].

[23] In all fairness, we notice that if the AP becomes much more aggressive than STAs (e.g., $\beta<0.5$ see definition d20), the a5 is not completely verified, since the average value of the AP buffer results a little bit lower than its maximum value. Anyway, we experienced this kind of problem only in case of pathological MAC level unfairness. We succeed in reproducing these pathological situations only by means of simulation, but we never found these pathological situations using our real equipments (e.g., Cisco and Linksys as APs and DLink and INTEL Centrino as WLAN cards).

Appendix IV ALTERNATIVE COMPUTATION OF THE SYSTEM IN LOSSLESS AP CASE

In this appendix we presents another approach to solve the system Ω (presented in Section 2.3) that may help to better understand the parameters relationships involved in the model.

We strive in making linear the equation system Ω, by opportunely eliminating the *max* operator from Eq. (9). The elimination of the *max* operator is done in two steps: i)we derive a sufficient and necessary condition for which Q_i>0 (i.e., Qup_i>0 and Qdw_i >0); ii)we use this condition in an iterative algorithm, which is based on an indirect proof approach.

This said, to address the problem of building and solving the system Ω we resort the iterative algorithm presented in Table 8, which is based on an indirect proof approach: in each iteration we assume that a set of queues are empties (i.e. Q_i=0) and the other are backlogged, we build and solve the system *Ω* controlling the backlogged conditions (Eq. (8)) for each queue.

```
Set the equation $Qup_i$= 0 for each i;
Set the equation $Qdw_i$= 0 for each i;
Set the equation Qap as indicated by the (14)
Compute linear system Ω and determine $Qup_i$, $Qdw_i$ and Qap
while (at least one $Q_i$=$Qup_i$+$Qdw_i$ is equal to zero and (8) is verified)
    {
      for each $Q_i$ that verify the (8)
      {
        set the equation $Qdw_i$=$Qdw_i$+0.5 $Pown\_dw_i$ – $\chi dw_i$
        set the equation Qupi=$Qup_i$+2 $Pown\_up_i$ – $\chi up_i$
      }
      Set the equation Qap as indicated by the (14)
      Compute linear system Ω and determine $Qup_i$, $Qdw_i$ and Qap
}
```

Table 8 - Iterative algorithm solving the system Ω

If the backlogged condition is verified for a queue a-priori assumed as empty, this implies an “absurd” and in the next iteration we will consider this queue as not empty. The iteration end when all the queues considered as empty do not satisfy the backlogged conditions, i.e. when none absurd occurs.

The solution of the system Ω has a complex analytical generalization but, anyway, it leads to the numerical values of the STA and AP average buffer occupancy.

Appendix V PROOF OF THE EFFECTIVENESS OF THE VBN ARCHITECTURE

In the following we derive a simple analytical model which proofs the effectiveness of the VBN architecture reported in Figure 34. We use the same assumption and definition previously adopted in Section 2.3.

In the CFTC framework, the network bottleneck is the VBN. It follows that, in a given instant the most of circulating packets are stored within the VBN. Starting from this consideration, we further assume that all circulating packets are stored within the VBN; i.e., we neglect the fact that a small number of packets may be in the other network pipes. Therefore, the VBN STA queues (see Figure 34) are not-empty and the *DRR* reserves to each VBN STA queue a capacity equals to $(C\text{-}\varepsilon)/M$.

Let us define as *VBN i-round*: *the time interval needed to send out all the packets buffered at the VBN queue of the i-th STA (hereafter indicated as i-th VBN STA queue) since the start of the round itself.* The *k*-th *VBN i-round* start at time t_k.

During each *VBN i-round* both TCP segments and TCP ACK are drained from the *i*-th VBN STA queue. For each TCP ACK drained in a round, two TCP segments will be queued in the next round. For each couple of TCP segments drained in a round, one TCP ACK will be queued in the next round. It follows that the number of segments ($Nseg_i(k+1)$) holds by the *i*-th VBN STA queue at the start of the ($k+1$)-th *VBN i-round* can be written as,

$$Nseg_i(k+1) = 2 \cdot Nack_i(k) \tag{27}$$

where $Nack_i(k)$ is the number of TCP ACKs holds by the *i*-th VBN STA queue at the start of the *k*-th *VBN i-round.*

By averaging the (27) and assuming that the involved processes are stationary we obtain that:

$$\frac{E[Nseg_i]}{E[Nack_i]} = 2 \tag{28}$$

In addition, since the TCP connections are operating without segment loss, the following equality holds;

$$E[Nseg_i] + 2 \cdot E[Nack_i] = Nup_i \cdot W_i + Ndw_i \cdot W_i \tag{29}$$

Form (28) and (29) it is possible to derive that,

$$E[Nseg_i] = \frac{1}{2}(Nup_i \cdot W_i + Ndw_i \cdot W)$$
$$E[Nack_i] = \frac{1}{4}(Nup_i \cdot W_i + Ndw_i \cdot W) \qquad (30)$$

The average delay *Dvbn*$_i$ experienced by a packet of the *i*-th STA in the *VBN i-round* can be expressed as:

$$Dvbn_i = \frac{E[Nseg_i] \cdot Lseg + E[Nack_i] \cdot Lack}{(C-\varepsilon)/M}$$
$$RTT_i = 2 \cdot \frac{E[Nseg_i] \cdot Lseg + E[Nack_i] \cdot Lack}{(C-\varepsilon)/M} \qquad (31)$$

where *Lack* and *Lseg* are the IP packet size in bits of a TCP ACK and of a TCP segment, respectively. Considering that the VBN queue of the *i*-th STA is crossed both by segments and acknowledge, it follows that the round trip time *RTT*$_i$ of the TCP connections of the *i*-th STA is equal to two times *Dvbn*$_i$.

From (29) and (31) we can derive the overall goodput of the *i*-th STA measured in bit per seconds as,

$$GPsta_i = \frac{(Nup_i \cdot W_i + Ndw_i \cdot W_i) \cdot Lpayload}{RTT_i} = \frac{Lpayload}{Lseg} \cdot \frac{2 \cdot Lseg}{2 \cdot Lseg + Lack} \cdot \frac{C-\varepsilon}{M} \qquad (32)$$

where, *Lpayload* is the length of the TCP payload measured in bits.

We observe that *GPsta*$_i$ is independent of the index *i* and, consequently each STA obtain the same goodput, *quod erat demonstrandumas*.