REVISION NOTE

Manuscript ID COMNET-D-10-540 entitled "Per-Station Throughput Fairness in a WLAN Hot-Spot with TCP traffic " which you submitted to the Elsevier Computer Networks journal

Authors:
A. Detti, N. Blefari-Melazzi, I. Habib, A. Ordine

FOREWORD

The authors are obliged to the reviewers for the time spent in revising the paper. In the following pages, we describe in detail how their concerns have been addressed. Some of the requests of the reviewers imply adding a lot of material in the paper. For space limitations we added this material to the extended version of our paper, which is available on line [17] and we referenced such additions in footnotes in the main paper. We can add this material in the main paper if the Editor agrees or if the reviewers suggest what we can remove from the current paper to make room for the new text.
ANSWERS TO THE COMMENTS OF REVIEWER #1
AND RELATED ACTIONS ON THE PAPER
(on a point by point basis)

Comment 1:
My main concerns are related to the authors' claim that their proposed approach "is easy to deploy in real systems". Indeed, in common deployments of WLANs (ESS - Extended Service Set), a set of APs and a gateway are connected to an (or even multiple) Ethernet switch. This scenario raises some questions:

how can the Linux box intercept all the traffic between each AP and the gateway? In the considered scenario, there is a back-to-back connection between the gateway and the Linux box and between the Linux box and the AP. In a real system, this is not possible, since APs (and the gateway) are directly connected to a switch.

Reply 1:
The scenario suggested by the reviewer is very common; indeed more than one AP may be placed behind a switch. In this case, a single Linux box can be placed between the gateway and the switch. The Linux box runs a “bank” of VSBs, where each VSB serves the traffic of a single AP. To classify the traffic belonging to a given AP, the Linux box retrieves from each AP the IP addresses (or the MAC addresses) of the STAs associated to that AP and configures a preliminary set of TC FILTERs, so as to send packets coming from a given AP (and going to that AP) to the proper VSB of the bank. From our experience, all APs allow to retrieve information on associated STAs, (e.g. via SNMP, HTTP, or by parsing HTML AP pages).

For instance, we retrieved association data from our Cisco Aironet access point; Cisco APs support the CISCO-DOT11-ASSOCIATION-MIB that contains information both on MAC and on IP addresses of associated STAs (OID name ciscoDot11AssociationMIB).

In the revised version of the paper we added the footnote 3, which summarizes this reply.

Comment 2:
how scalable is the proposed solution? In an ESS, the number of stations may be in the order of hundreds, hence hundreds of rules and filters should be installed on the Linux box. How would the performance of the system degrade? Could the Linux box become a bottleneck?

Reply 2:
To answer to this interesting question we have assessed the scalability of our Linux Box (equipped with an Intel Centrino 1.4 Ghz) in the test-bed depicted below.

We have a STA where we have setup 120 logical interfaces (wlan0:1, wlan0:2, wlan0:3… wlan0:120). Each interface has a different IP address. On the STA there is an IPERF server which receives TCP segments coming from IPERF clients; i.e. we consider downstream traffic. IPERF clients are located on a wired PC behind the gateway. Each IPERF client send packets toward a specific interface of the STA and the number of IPERF clients is equal to the number of interfaces.

Within the Linux box, we setup a number of filters and HTB classes equal to the number of interfaces. We observe that such a scheduler configuration is equal to the one that the Linux box would have in case of several STAs (up to 120), with a single IP address per STA. Therefore, as regards the processing load of the
Linux box, the case of a STA with 120 IP addresses is equal to the case of 120 STAs with a single IP address per STA.

At the beginning of the test, the STA has only one interface (i.e. wlan0:1), there is only one active IPERF connection toward such interface and there is only an HTB class/filter. Then, every 5 seconds a new interface is added, a new IPERF connection toward this interface is started and a new HTB class/filter is added to the scheduler. We measure the overall goodput of the TCP connections during the time and the obtained results are reported in the figure below. We observe that the overall goodput is quite constant (with the exception of a small initial decrease discussed afterwards); therefore, the Linux box is insensitive to the number of HTB classes/filters and it is not a bottleneck. If the Linux box were a bottleneck, we would have measured a monotone and significant decrease of the performance, as the time increase, i.e. at the increase of HTB classes/filters.

We observe that the overall goodput has a very limited initial decrease. This occurs because the overhead reduction yielded by the delayed-ack mechanism tends to vanish. When the number of TCP connections is small, we have one ACK every two segments, but when there are many TCP connections, the goodput of each TCP connection is so small that the timeout of the delayed-ack mechanism always expires; consequently, we have one ACK for each TCP segment. A greater number of ACKs implies a lower availability of wireless resource for TCP segments, i.e. a lower TCP goodput.

Finally, we observe that the negative spikes of the plot are likely due to synchronization phenomena of the TCP, i.e., some TCP connections concurrently go in time-out.

Since the Linux box scalability does not seem to be an issue, we have not added any related discussion in the revised version of the paper, also because we do not have space (to respect editor’s space limitations). However, if the reviewer prefers we can add this material to the extended version of our paper [17] which we will keep available on line, and a footnote referencing this material in the main paper.

![Graph showing goodput over time](image.png)

**Comment 3:**

the proposed solution requires a rule to be installed on the Linux box for each station. Are such rules pre-configured statically or is there a means to configure them dynamically in an automated fashion?

**Reply 3:**

We give two answers to this comment.

First, the previous scalability test showed that installing hundreds of HTB classes/filters in the Linux box does not cause any problem; therefore, we could pre-install in the Linux-box a number of HTB classes/filters big enough so as to accommodate the maximum number of IP addresses that the WLAN can manage; e.g. the range of IP addresses configured in the DHCP server.
Secondly, our first implementation of the Linux box was not the one reported in the paper (i.e., the one that exploits Linux Traffic Control); our first implementation supported the dynamic insertion of STA queues in a Deficit Round Robin scheduler. Such first implementation consisted of a user-space multi-thread C++ program. To handle the operations required to receive and to send packets, at user-space, we used the LIBIPQ library. The code of the Deficit Round Robin (DRR) is the one available in the Network Simulator 2 project, with some modifications required to handle the packet format of LIBIPQ and to classify the traffic as a function of the STA IP address. The user-space VSB automatically inserts a new queue in the DRR, whenever a new IP address is detected.

We abandoned this kind of implementation because the use of the Linux TC tool makes the scheduling framework more easily programmable. However, we provide the reviewers with the code of this implementation (http://netgroup.uniroma2.it/Andrea_Detti/VSB/vsb-libipq.tar.gz) to show that it is viable to devise a scheduler that automatically manages the number of queues, it is only a programming effort without any impact neither on the feasibility nor on the performance of the VSB; for this reason we do not discuss this issue in the revised version of the paper. If the reviewer wants us to add a dynamic implementation we could publish a new software on the web [18], that upgrades and improves our latest VSB software.

For instance, we could develop a user-space C++ controller based on LIBIPQ that controls the active STAs and inserts and removes HTB classes/filter.

**Comment 4:**
Finally, I would like to see how the system performs in case the TCP connections are established with servers in the Internet.

**Reply 4:**
We have added this scenario in our testbed (see section 5.4 of the revised version of the paper).
ANSWERS TO THE COMMENTS OF REVIEWER #2
AND RELATED ACTIONS ON THE PAPER
(on a point by point basis)

Comment 1:
The current wireless capacity estimator is based on the assumption that the network nodes have empty queues, thanks to the VSB. Therefore, the authors consider a fixed RTT threshold for inferring about the network congestion status. I suggest to also consider a possible solution in presence of stations involved in UDP sessions. In this case, since the VSB cannot control the queue of an UDP source located on the contending stations, the RTT threshold should depend on the number of UDP contending nodes and on their employed rates. Is it possible to make this parameter adaptive, by complicating the network-state estimation process? Are the VSB performance critically affected by the RTT threshold setting in absence of UDP sources? Although the author state that the Wireless Capacity Estimator is not the major goal of their contribution, the VSB should depend critically on such a component.

Reply 1:
We are very grateful to the reviewer because he has motivated us to study what happens in presence of UDP traffic. We setup a “TCP&UDP” testbed formed by two STAs. A STA supports TCP traffic while the other STA supports UDP traffic. As regards the direction of the traffic, we analyze all the 4 different cases: TCP downstream - UDP downstream, TCP upstream - UDP downstream, TCP upstream - UDP upstream and TCP downstream - UDP upstream. The findings of this study are reported in Appendix IV of the extended version of the revised paper [17] and not in the paper itself for space limitations (we did add a footnote referencing this material in the main paper).

Comment 2:
Since the authors state that the VSB can be easily modified for enforcing other fairness criteria, it could be interesting to discuss, at least in general terms, how to make the VSB "programmable" to different goals and working conditions. What happens when stations activate/de-activate dynamically? Is a new queue allocated/de-allocated on the basis of the filtered SYN/FIN packets or does the current VSB implementation only assume a pre-configured number of stations M?

Reply 2:
As regards the VSB programmability, we added Appendix V in the extended version of the revised paper, in which we give an example of how the Linux VSB can be programmed to enforce different types of fairness (and we added a footnote referencing this material in the main paper). As regards “What happens when stations activate/de-activate dynamically?” this comment is equal to the comment #3 of reviewer #1, so we refer this reviewer to reply #3 to reviewer #1.

Comment 3:
As far as concern the fairness model, the authors consider a static beta value, that has been experimentally measured in a specific scenario (1 station contending with the AP). However, if the difference on the AP/STA channel access probability depends on the card implementation and/or contention settings, such a parameter could change according to the average number of contending stations. For example, if the difference is due to heterogeneous values of the minimum contention window, the average value of the backoff counter (which impact the channel access probability in greedy conditions) could change as a function of the collision probability experienced in the network.
Reply 3:
Yes, we definitively agree with the reviewer. However, to simplify the model we assumed a static value of beta. We should have inserted in the model an “inner” procedure to dynamically evaluate beta. Nevertheless, beta may differ from 1 due to a lot of “mysterious” technical causes. For instance, different values of CWmin; or a different way to handle backoff, e.g. in terms of slot or of continuous time; bad implementation of the firmware timing; other implementation-dependent factors etc. Therefore, it seems very difficult to devise a model for a dynamic beta, indeed this parameter can depend on many causes. In all fairness, we did not want to introduce beta in the model at all, but unfortunately our AP and DLINK WiFi cards have a different access probability and so we needed to model this behavior with beta in order to have a match between theoretical and experimental results. Indeed, our model is accurate.
In the revised version of the paper, we stress in footnote 7 that our model results are tight in case of static beta: “Moreover, the value of $\beta$ may also vary as a function of different WLAN condition (e.g., collision probability), even if this did not happen during our tests. If $\beta$ varies during the test, modeling results may slightly differ from the measurements, since in the model we assume a static value of $\beta$."

Minor Comment n.1:
At pg. 11, the number of contending stations M has not been defined.

Reply to minor Comment n.1:
We added this definition in the revised version of the paper. “…HTB leaf class with guaranteed rate equal to C/M, where M is the number of STAs.”

Minor Comment n.2:
All results refer to the 802.11b PHY layer, which could be considered outdated.

Reply to minor Comment n.2:
We are truly sorry about this but our RALINK hardware does not perfectly support the 54 Mbit/s rate on Linux. Anyway, the phy-rate does not alter the fairness results, as we stated in property 4 “The MAC physical transmission rate does not affect the fairness performance”.

Minor Comment n.3:
The considerations in the footnote number 9 (about the different assumptions on AP lossless or lossy behaviors) are important and could be explicitly discussed with the fairness model general assumptions.

Reply to minor Comment n.3:
We moved this text from footnote to main text, in the revised version of the paper.