SplitAP: Leveraging Wireless Network Virtualization For Flexible Sharing Of WLANs

Gautam Bhanage†, Dipti Vete†, Ivan Seskar† and Dipankar Raychaudhuri†
†WINLAB, Rutgers University, 671 Rt1 South, North Brunswick, NJ 08902, USA
{gautamb, dvete, seskar, ray}@winlab.rutgers.edu

***Abstract*— Providing air-time guarantees across a group of clients forms a fundamental building block in sharing an access point (AP) across different virtual network providers. Though this problem has a relatively simple solution for downlink group scheduling through traffic engineering at the AP, solving this problem for uplink (UL) traffic presents a challenge for fair sharing of wireless hotspots. Among other issues, the mechanism for uplink traffic control has to scale across a large user base, and provide flexible operation irrespective of the client channel conditions and network loads. In this study, we propose the SplitAP architecture that address the problem of sharing uplink airtime across groups of users by extending the idea of network virtualization. Our architecture allows us to deploy different algorithms for enforcing UL airtime fairness across client groups. In this study, we will highlight the design features of the SplitAP architecture, and present results from evaluation on a prototype deployed with: (1) *LPFC* and (2) *LPFC*+, two algorithms for controlling UL group fairness. Performance comparisons on the ORBIT testbed show that the proposed algorithms are capable of providing group air-time fairness across wireless clients irrespective of the network volume, and traffic type. The algorithms show up to 40% improvement with a modified Jain fairness index.**

I. Introduction

The advent of pervasive wireless systems in the form of inexpensive handheld devices is expected to lead to an ever increasing deployment of wireless hotspots [12]. With more and more ISPs aiming to provide services at locations such as airports, cafes and common shopping areas, differentiation in the quality of service provided on shared hardware for wireless ISPs provides a substantial challenge. A mechanism is required to ensure that this access point (AP) sharing will work across a wide range of client hardware, while providing each user group (clients belonging to a single ISP) with aggregate air-time commensurate to the revenue contract of the ISP with the wireless infrastructure provider. Apart from providing baseline fairness in air-time across groups, other requirements for sharing WLAN access point hardware across different ISPs include: (1) Different broadcast domains, (2) Different levels of security, (3) Support different protocols above a basic L2 connection, (4) Ease of deployment, and (5) Minimum bandwidth loss for resource partitioning.

To solve this problem we propose the SplitAP architecture that employs wireless network virtualization. Network virtualization is a concept derived from the server systems area of research which has recently been applied to network sharing. Virtualization is a mechanism that allows for seamless sharing of a particular resource by using three key features: *Abstraction, Programmability and Isolation*. We apply each of these features as shown in the Figure 1. *Abstraction* allows the users of the system to use our architecture with minimal changes to the client hardware or software. As shown in the Figure, we use virtual access points [4] supported by most commodity AP hardware to emulate the functionality of two different physical APs (ISP1, ISP2) with a single physical AP, thus allowing us to use the client MAC protocols and hardware unchanged. We provide *programmability* in the setup by allowing the person deploying the hardware to allocate different UL air-time quotas for individual virtual access points. Finally, *isolation* across groups of wireless users is provided through air-time control at the clients based on the information provided by the SplitAP controller running at the AP. Since downlink air-time fairness has been studied previously [6], and a spate of recent applications such as those supported by web 2.0 [3], peer-to-peer file sharing [18], and video conferencing have resulted in significantly increased uplink air-time usage, we specifically address the problem of uplink air-time control across the virtual networks formed by wireless user groups. Through the use of a SplitAP prototype discussed in the paper, we will show the performance of our sample algorithms for providing uplink air-time fairness across user groups, while providing all of the features discussed above.

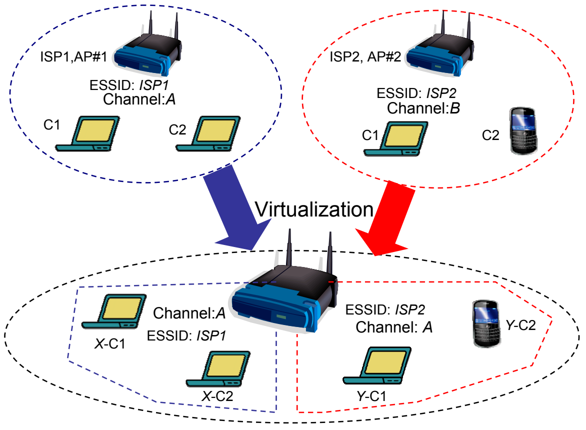


Fig. 1. A single wireless access point emulating multiple virtual access points. Clients from different networks associate with corresponding VAPs though they use the same underlying hardware.

Specifically the contributions of this study are:

1) We propose, design and implement the SplitAP software

architecture based on the extension of the virtual access point functionality for sharing a single physical AP across groups of users.
2) We design and evaluate the *LPFC* and *LPFC+* algorithms for group UL air-time control using our SplitAP setup on commercial *off-the-shelf* hardware.
3) Extensive evaluation is performed to show that the results obtained on our infrastructure are as per the requirement while achieving the system performance with minimal overhead.

Rest of the paper is organized as follows. In Section II we discuss related work on previous approaches to providing uplink air-time fairness in WLANs. Section III discusses the problem of providing uplink air-time fairness across user groups, and presents the design of our SplitAP architecture. Section IV presents a discussion on the two sample algorithms evaluated with the SplitAP framework. Section V presents the results from the system, and finally, Section VI discusses the conclusions and future work.

II. Related Work

Among AP based infrastructures, the *DenseAP* architecture proposed in [16], describes a mechanism for sharing airtime by managing handoffs across APs. Another setup to share downlink air-time has been discussed for WiMAX radios in [6]. Our *SplitAP* setup specifically deals with the problem of providing an architecture for sharing UL air-time of a single AP across multiple WLAN user groups. In terms of the methodology itself, a comparison of wireless virtualization approaches is presented in [15]. However, it does not address the problem of fair sharing of UL air-time across client groups.

In the domain of air-time fairness, a body of work [14], [10], [7], [5] discusses the use of EDCA parameters such as contention windows and transmission opportunities for controlling airtime usage across clients. The study in [14] attempts to ensure fairness across competing uplink stations with TCP traffic using EDCA parameters. Time fair CSMA protocol proposed in [10] controls minimum contention window size to achieve estimated target uplink throughput for each competing station in multirate WLAN. In [5] authors suggest that in a proportional fair allocation based on $802.11e$ EDCA parameters, equal share of channel time is given to high and low bit rate stations and, as a result, high bit rate stations can obtain more throughput. Another study in [7] proposes two control mechanisms for airtime fairness, one using AIFS and the other using contention window size. The studies in [14], [10], [7], [5] are based on simulations.

One study in [17] proposes a Time Based Regulator system that achieves uplink air-time fairness by ensuring equal "long term" channel occupancy time for every node in the WLAN. Though this study presents results based on an implementation, it does not deal with the problems of clients sending traffic with different frame sizes, offered loads, and sharing of airtime across user groups. The TWHTB system discussed in [8] uses information on current channel quality to the respective station associated with AP to schedule downlink transmission to that particular station by limiting frame transmission rate. However, this scheme does not take into account Uplink flows and corresponding traffic variations. Another study discussed in [11] discusses an approach where each station monitors the number of active stations and calculates the target access time based on this information. The study uses sniffing on the client side, while also requiring modification of NAV field in the MAC header, and results are based on simulations.

In addition none of these studies address the problem of enforcing *client-group* UL airtime fairness which is addressed by algorithms run on our SplitAP setup.

III. SplitAP Design Overview

Throughout this study we use the notion of *slices* to refer to the resources allocated to a group of users belonging to a single ISP. The terms *groups* or *slices* will be used interchangeably in further discussion. Our infrastructure will try to enforce fairness in uplink (UL) airtime usage across slices, thus allowing individual ISPs to fairly share the underlying WLAN hardware and the corresponding channel. We begin with a formal definition of the problem of sharing UL airtime across a group of users, followed by a conceptual description of our virtualization based design. Eventually, we will discuss the details of the algorithms used for UL airtime allocation.

A. Group Uplink Airtime Fairness: Problem Statement

Consider a set of M client groups (slices) with each group S_i having N_i clients. Let the fraction of UL air time allocated for every slice $S_i \in M$, be denoted by W_i. W_i for each slice is decided during the time of deployment of the infrastructure and can be dependent on a wide range of criterion like pricing, importance of the group and so on. If ϕ_j^i denotes the measured UL air time consumed by the client $j \in S_i$ slice, the fraction of UL air-time used by every client associated with the access point is calculated as C_j^i:

$$C_j^i = \frac{\phi_j^i}{\sum_{p=1}^{M} \sum_{q=1}^{N_i} \phi_q^p} \tag{1}$$

The condition of group fairness requires that, the total measured UL airtime for all clients within a slice S_i is limited to W_i:

$$\forall \{i \in M\}, \quad \sum_{j=1}^{N_i} C_j^i \leq W_i \tag{2}$$

The above condition should be fulfilled while placing no limitation on the individual values of C_j^i i.e all nodes within a single slice S_i should be able to share the UL airtime fairly, independent of the usage on other slices. Hence, in the worst case every client should be able to utilize UL airtime $0 \leq C_j^i \leq W_i$ as long as the Equation (2) is not violated and all clients within S_i share the available UL airtime fairly.

Qualitatively summarizing the constraints of the slice/group fairness mechanism: (1) Flexiblity: If the channel usage is below saturation, and there are no hard guarantees, each client should be able to access the entire available channel time for the slice, (2) Within a group: Sharing of UL airtime should be

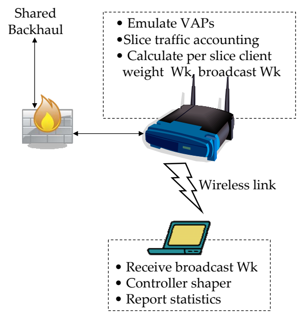


Fig. 2. A single wireless access point emulating multiple virtual access points. Clients from different networks associate with corresponding VAPs though they use the same underlying hardware.

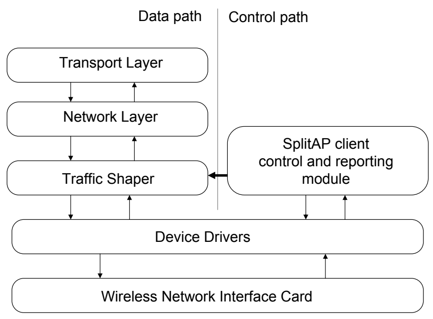


Fig. 3. Network stack at the wireless client associating with the SplitAP infrastructure.

fair and equal, (3) Scalable: Should work with a large number of clients without significant control overheads, (4) Adaptable: Should be able to comfortably adapt to changing environment with dynamic addition or removal of wireless clients, the network load, protocol type and the channel conditions for individual clients. Hence, to allow deployment of algorithms that will be able to realize such a group airtime fairness mechanism, our SplitAP infrastructure will need to provide all needed control and measurement features while being transparent to the users of the system.

B. Virtualization Based Design

We will now discuss how each of the virtualization features are implemented as a part of our SplitAP architecture.

Abstraction: We employ and extend the functionality of virtual access points which are available as a standard feature on commercial access points for emulating multiple virtual access points on a single physical access point while operating on the same wireless channel [4]. Using this feature the physical AP will be able to broadcast beacons for independent virtual networks (ISPs). Hence clients belonging to different ISP slices can see the ESSID of their ISP and associate with it, thereby making client side connectivity transparent and simple.

Programmability: Each of the ISPs should have independent control of settings in their network. Using virtual access points, we can set different features per WLANs such as different security policies, broadcast domains, IP settings, independent control of MAC settings such as aggregation and *802.11e* based WMM parameters.

Isolation: Isolation across virtual networks (client groups) is a fundamental requirement for supporting multiple networks and will be the main topic discussed in this paper. Ideally, this could be done through a strict TDMA scheduler across the virtual networks. However, such a scheduler would require a large change in the MAC mechanism of the clients, thus making them completely incompatible with other 802.11 based commercial access points. The SplitAP mechanism proposed in this paper is an incremental design to the existing *802.11* framework and is currently capable of existing as a stand alone entity outside of the driver.

The functionality in our system is split as shown in the Figure 2. The SplitAP controller at the AP is responsible for emulating the virtual access points, accounting of traffic by client groups, and determining the weights of UL airtime for each group. The client software is responsible for enforcing the commands broadcasted by the controller and reporting usage statistics like the physical layer rate and the average packet size reported by the client interface. The remaining discussion will focus on the implementation of individual components, followed by a brief overview of our algorithms for providing uplink airtime fairness across the ISP slices.

C. SplitAP Controller

The access point infrastructure runs a multi-threaded ruby controller that performs the actions described in Algorithm (1). In the controller, $sliceID$ is a unique identifier used for identifying independent slices owned by different ISPs. The algorithm computes slice UL airtime usage $time[sliceID]$ for every $sliceID$, by iterating and determining the UL airtime usage reported by individual clients within every slice $sliceID$. Based on this estimate, it determines the offset of the actual slice utilization from the allocated UL airtime fraction. If this offset is greater than a threshold (Θ), the AP controller broadcasts[1] $C_{sliceID}$ the maximum UL airtime fraction that can be consumed by any individual client within the slice $sliceID$. The value of $C_{sliceID}$ is always chosen as inversely proportional to the UL airtime utilization for that slice. This fraction of channel time is calculated based on the previously broadcasted value and the corresponding slice utilization. *LPFC* and *LPFC+* algorithms discussed later are two means of calculating $C_{sliceID}$ based on current UL airtime utilization numbers and or the number of associated clients.

D. Client Plugin Design

In the current design, the client needs to install an application that allows the user to connect to a SplitAP based wireles service provider. Eventually, to make this application platform independent, it could be implemented as a web browser *plugin*

[1]UDP broadcast is deliberately used as a means of sending $C_{sliceID}$ to clients to limit control traffic, since the number of control messages are now dependent on the number of slices rather than number of clients. Ideally, these $C_{sliceID}$ will be included in the beacons of individual virtual access points, thereby eliminating the need for a separate signalling mechanism.

Algorithm 1: The SplitAP controller running at the AP that monitors slice usage and correspondingly broadcasts slice weights to clients.

```
$W_{0..M} = init\_slice\_weights()$
while True do
    foreach $sliceID$ = getNextSlice() do
        $time[sliceID]$ = 0
        foreach $clientID$ = getNextClient($sliceID$) do
            $rate$ = getPhyRate($clientID$)
            $bytes$ = getDataVol($clientID$)
            $cl\_time = \frac{bytes \times 8}{rate}$
            time[sliceID] + = $cl\_time$
        end
        $\delta$ = time[sliceID] - $W_{sliceID}$
        if (abs($\delta$) > $\Theta$) then
            $C_{sliceID} = getWt(sliceID, slice\_wt, \delta)$
            $broadcast(sliceID, C_{sliceID})$
        end
    end
end
```

that controls client's UL traffic based on commands from the controller. The client software stack in the current SplitAP architecture is as shown in Figure 3. The SplitAP client control and reporting module is responsible for two functionalities: (1) Determining and reporting client side parameters such as physical layer rate (through access of the rate table maintained in the driver), and average packet sizes by querying the *proc* filesystem or using the driver statistics. (2) Converting the maximum airtime limit enforced by the SplitAP controller to a rate value, and accordingly controlling the shaping module to rate limit the client. The shaping module is implemented by using the Click [13] modular router that transparently controls outbound traffic from the interface.

IV. Algorithms For Deployment With SplitAP

Our SplitAP design offers a convenient way to deploy different algorithms on the AP for controlling uplink airtime across slices. Each of the algorithms discussed in this section are ways to implement the *getwt()* function discussed in Algorithm (1) and provide the value $C_{sliceID}$, which is the maximum airtime that can be consumed by any client in Slice $S_{sliceID}$.

A. Algorithm(1): LPFC

This is a simple linear proportional feedback control (LPFC) based algorithm that uses a dynamic estimate of the number of clients associated with the AP to calculate the $C_{sliceID}$. Information on the number of clients associated with the AP is available in the SplitAP controller through querying of the *proc* interface on the AP. The algorithm calculates $C_{sliceID}$ simply by determining current number of clients in the slice $S_{sliceID}$ and proportionally splitting the available (quota of) airtime $W_{sliceID}$ among the number of clients $N_{sliceID}$ within the slice. The SplitAP architecture allows this corrected $C_{sliceID}$ to be broadcasted every one second or at another interval desired by the ISP using the slice.

B. Algorithm(2): LPFC+

Instead of generating and broadcasting the $C_{sliceID}$ purely based on the slice UL airtime quota and number of clients in the slice, the LPFC+ algorithm relies on monitoring the current UL airtime utilization for the slice, which is available through the SplitAP client reports and appropriately controlling $C_{sliceID}$. The algorithm selects $C_{sliceID}$ in such a way that even if the offered load by clients in a slice is not the same, it allows the clients to increase traffic, by increasing $C_{sliceID}$ until the UL airtime quota for the slice is reached. If the quota is exceeded (or under-utilized), the LPFC+ controller proportionally reduces (or increases)$C_{sliceID}$, the maximum airtime that can be used by any client in the Slice $sliceID$. As with the LPFC algorithm, the $C_{sliceID}$ can be broadcasted every one second or at any other value desired by the ISP owning the slice.

V. Experimental Evaluation

All experimental results presented in this evaluation are based on the clients with Atheros 5212 chipsets, and using Madwifi 0.9.4 [2] drivers. The clients are all operating in the 802.11a mode with a frame size of 1024bytes, and 54Mbps physical layer rate unless mentioned otherwise. Traffic is generated with the Iperf tool [1]. We begin with a brief definition of the metrics used, followed by baseline performance of the *LPFC* algorithm and a comparison with *LPFC+*.

A. Metrics

Preliminary evaluations with a small number of clients will be based purely on comparison of UL airtime allocated to individual slice. Further, in our evaluations, we modify and use the Jain fairness index [9] for determining weighted UL airtime fairness across flows and flow groups.

Modified Jain Index: Let the sum of fraction of channel time used by all clients in slice k be denoted as C_k.Then,

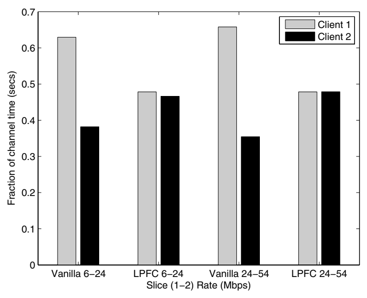
$$I = \frac{\left(\sum_{k=1}^{N} C_k\right)^2}{N \times \sum_{k=1}^{N} C_k^2} \quad (3)$$

The fairness index (I) determines the global variation in channel utilization across slices. We further scale the airtime by slice quotas to evaluate fairness under saturation with different slice weights, while also accounting for performance deterioration due to bad channel quality.

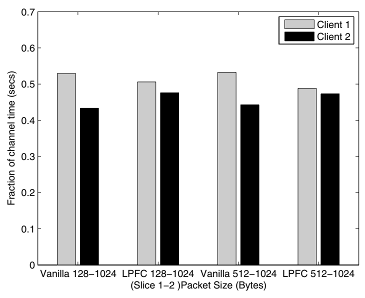
B. Baseline Performance With LPFC

To measure the baseline performance with the *LPFC* algorithm, we consider a setup with two clients on different slices sending UDP UL traffic.

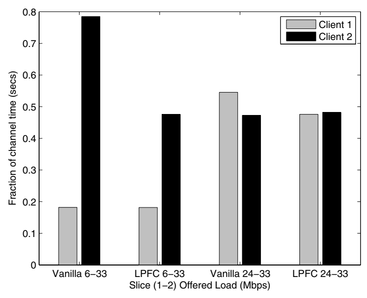
Varying Transmission Rates In the first experiment, we vary transmission rates of the two clients on Slice 1 and



(a) Varying rate



(b) Varying packet size



(c) Varying load

Fig. 4. Baseline results for comparison of performance with and without the the SplitAP setup with the *LPFC* algorithm. Results indicate performance with two clients on different virtual networks with varying physical layer transmission rates used with a UDP saturated offered load.

2 as shown on the x-axis in Figure 4(a). We observe that, in the vanilla case (without the SplitAP mechanism running the *LPFC* algorithm), the air-time used by the two clients are inversely dependent on the transmission rates. This is a result of statistical multiplexing of packets by the CSMA MAC operating as a part of the 802.11 DCF mechanism. To alleviate this problem, the SplitAP framework controls the shaping of traffic at the source to consequently provide proportional fair channel usage across clients. As shown in the results in Figure 4(a), we are able to control air-time provided to each client in desired proportion. Sample results are provided for a $50-50$ percent air-time sharing across the two clients.

Varying Packet Sizes Now we will vary the packet size of the uplink traffic from each client to check its impact on the overall sharing of air time at the access point for the two clients. As seen in the results in Figure 4(b), the air-time consumed at the access point without the use of our scheme (vanilla) is directly dependent on the size of the packets used by the uplink traffic. Typically, this results from a statistical multiplexing of packets over the air. However, using our SplitAP infrastructure with the *LPFC* algorithm we are able to control uplink traffic in direct proportion to the air time usage by each client. Our scheme accounts for the extra air-time spent in channel accesses and PHY/MAC overheads with smaller packet sizes resulting in fair sharing across the clients and thus virtual networks. As before we observe, that our infrastructure allows control of air-time across the clients in a preset $50-50$ percentage. In later experiments this percentage will be changed.

Varying Offered Loads In this experiment, we vary the offered loads across the two clients. Combinations of offered loads used across the clients are as shown on the x-axis in the results in Figure 4(c). The maximum offered load is limited to 33Mbps because the channel saturates at that value[2] of the offered load when the physical layer rate is 54Mbps. We observe that the *LPFC* algorithm limits airtime of slice 1 (with 33Mbps physical rate) even though the other client is not using its share. Even though the *LPFC* scheme is conservative, it limits airtime of Slice 2, to ensure better fairness as compared to the vanilla case with no control.

[2]The channel saturates at a slightly higher value than normal since the Madwifi drivers use fast framing optimizations to improve performance within allocated txops. However, this does not affect our evaluation since it is enabled in all measurement cases.

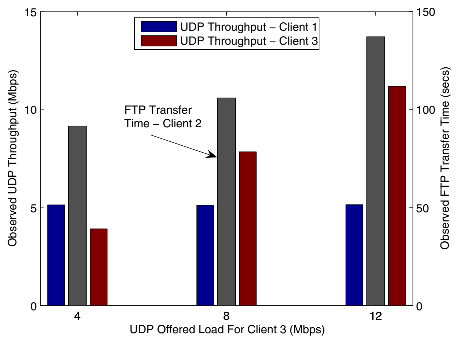


Fig. 5. TCP and UDP co-existence in a single slice with *LPFC+*. Constant UDP traffic of 5Mbps is supported by slice 1, while the Client 2 with FTP transfer and the client 3 with varying UDP loads share the slice 2.

C. Improvement With LPFC+

Since the LPFC+ algorithm allows the allocation of slice weights such that within a slice we may have varying utilization by independent clients, such a mechanism allows for fair co-existence of transport protocols with different requirements. In this experiment we have two slices: Slice 1 has a client sending constant UDP uplink traffic, while the Slice 2 has two clients. The first client in Slice 2 is sending varying amount of UDP uplink traffic, while the other client in Slice 2 is transfering a 200MB file with a FTP file transfer. Results from this experiment are as shown in Figure 5. We observe that the client on slice 1 is not affected despite one of the clients on Slice 2 using UDP traffic. We also observe, that the clients on Slice 2 share the UL airtime. When the UDP offered load is less at 4Mbps, the FTP transfer is faster and happens at an aggregate rate of 18.3Mbps. When the UDP offered load on the client increases, the FTP client reduces its rate, thereby requiring longer time for the FTP transfer completion. It is important to note that a similar performance could be achieved even by using *LPFC* instead of *LPFC+*. However, in that case

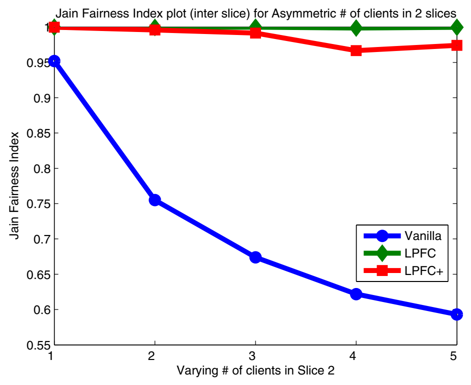


Fig. 6. Comparison of UL airtime group fairness for: LPFC, LPFC+, and a vanilla system without our SplitAP framework.

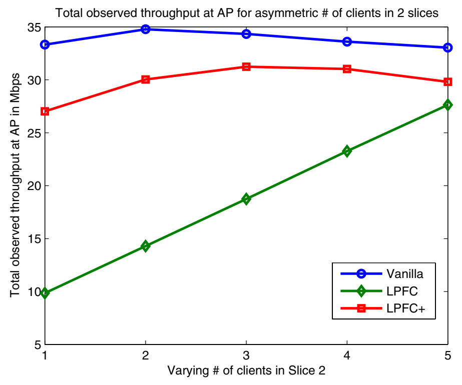


Fig. 7. Comparison of UL throughput for: LPFC, LPFC+, and a vanilla system without our SplitAP framework.

the FTP client on slice 2 would always be limited to a fixed uplink rate thereby resulting in a wastage of free bandwidth.

D. Comparison: LPFC Vs LPFC+

In a final experiment we consider a setup with two slices: Slice 1 has a single client pumping UDP UL traffic at saturation, while Slice 2 has 5 clients associated with it. For different experiments, varying number of clients $1 - 5$ on Slice 2 will send saturation UL traffic along with the client on Slice 1. In this case we consider the performance of both *LPFC* and *LPFC+* algorithms, as compared to that without our SplitAP setup (Vanilla). A comparison of the measured modified fairness index is as shown in Figure 6. We observe that the group fairness index I is always greater than 0.97 with the use of our infrastructure, while it falls down up to 0.6 in a vanilla system without our setup. The throughput measurements in Figure 7 show that the improvements in fairness are at the cost of a small decrease in net throughput with *LPFC+*, thus justifying the use of our scheme. The throughput performance with our *LPFC* scheme is less when lesser number of clients on Slice 2 pump traffic. This is because it sees 5 clients associated with the slice from the beginning, and presents a conservative estimate of $C_{sliceID}$ which results in lower throughput. The *LPFC+* scheme on the other hand dynamically measures airtime for every slice and adapts its $C_{sliceID}$ resulting in better performance. It cannot reach channel capacity since it keeps a 15% tolerance, but is able to divide the remaining airtime fairly.

VI. Conclusions And Future Directions

In this study we describe the SplitAP architecture that allows the operator to deploy a shared physical access point, which is capable of running different algorithms that control UL airtime across user groups. We demonstrate feasibility of the proposed architecture by implementing the *LPFC* and *LPFC+* algorithms on a prototype. Results obtained from the measurements on the ORBIT testbed show a significant improvement in the group airtime fairness, while resulting in marginal degradation of overall system throughput. Future directions include search for more efficient algorithms that can be deployed on the SplitAP framework.

Acknowledgment

Research supported in part by the US National Science Foundation Grant $\#CNS - 072505$.

References

[1] Iperf traffic generator. *http://sourceforge.net/projects/iperf/*.
[2] Madwifi driver. $http://www.madwifi.org$.
[3] Web 2.0 framework. *http://tinyurl.com/dqt86*.
[4] B. Aboba. Virtual access points, ieee document, ieee 802.11-03/154r1. *http://tinyurl.com/yjjkwpv*.
[5] A. Banchs, P. Serrano, and H. Oliver. Proportional fair throughput allocation in multirate ieee 802.11e wireless lans. *Wireless Networks*, 13(5):649–662, 2007.
[6] G. Bhanage, R. Daya, I. Seskar, and D. Raychaudhuri. VNTS: a virtual network traffic shaper for air time fairness in 802:16e slices, 5 2010.
[7] C.-T. Chou, K. G. Shin, and S. S. N. Contention-based airtime usage control in multirate ieee 802.11 wireless lans. *IEEE/ACM Trans. Networking*, 14(6):1179–1192, 2006.
[8] R. G. Garroppo, S. Giordano, S. Lucetti, and L. Tavanti. Providing air-time usage fairness in ieee 802.11 networks with the deficit transmission time (dtt) scheduler. *Wirel. Netw.*, 13(4):481–495, 2007.
[9] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, DEC, September 1984.
[10] T. Joshi, A. Mukherjee, Y. Yoo, and D. P. Agrawal. Airtime fairness for ieee 802.11 multirate networks. *IEEE Transactions on Mobile Computing*, 7(4):513–527, 2008.
[11] D.-Y. KIM, E.-C. PARK, and C.-H. CHOI. Distributed access time control for per-station fairness in infrastructure wlans. *Transactions on Communications*, Vol.E89-B(9):2572–2579, 2006.
[12] K. Knight. Jiwire: Wifi to become predominant connection for mobile users. *http://tinyurl.com/yjukaq9*.
[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, 2000.
[14] D. Leith, P. Clifford, D. Malone, and A. Ng. Tcp fairness in 802.11 e wlans. *Communications Letters.*, Vol.9(11):964–966, 2005.
[15] R. Mahindra, G. Bhanage, G. Hadjichristofi, I. Seskar, D. Raychaudhuri, and Y. Zhang. Space versus time separation for wireless virtualization on an indoor grid. In *proceedings of NGI*, pages 215–222, April 2008.
[16] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill. Designing high performance enterprise wi-fi networks. In *proceedings of the 5th USENIX NSDI conference*, pages 73–88, Berkeley, CA, USA, 2008.
[17] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate wlans. In *proceedings of USENIX ATEC*, pages 23–23, Berkeley, CA, USA, 2004. USENIX Association.
[18] M. Wagner. How iphone 3.0 may revolutionize the smartphone industry. *http://tinyurl.com/c4vkpa*.