Experiences With a Centralized Scheduling Approach for Performance Management of IEEE 802.11 Wireless LANs

Malati Hegde, Pavan Kumar, K. R. Vasudev, N. N. Sowmya, S. V. R. Anand, Anurag Kumar, *Fellow, IEEE*, and Joy Kuri

***Abstract*—We present a centralized integrated approach for: 1) enhancing the performance of an IEEE 802.11 infrastructure wireless local area network (WLAN), and 2) managing the access link that connects the WLAN to the Internet. Our approach, which is implemented on a standard Linux platform, and which we call ADvanced Wi-fi Internet Service EnhanceR (ADWISER), is an extension of our previous system WLAN Manager (WM). ADWISER addresses several infrastructure WLAN performance anomalies such as mixed-rate inefficiency, unfair medium sharing between simultaneous TCP uploads and downloads, and inefficient utilization of the Internet access bandwidth when Internet transfers compete with LAN–WLAN transfers, etc. The approach is via centralized queueing and scheduling, using a novel, configurable, cascaded packet queueing and scheduling architecture, with an adaptive service rate. In this paper, we describe the design of ADWISER and report results of extensive experimentation conducted on a hybrid testbed consisting of real end-systems and an emulated WLAN on Qualnet. We also present results from a physical testbed consisting of one access point (AP) and a few end-systems.**

***Index Terms*—IEEE 802.11 wireless networks, split-medium acces control (MAC) wireless local area network (WLAN) controllers, WLAN controllers, WLAN quality-of-service (QoS) management.**

I. Introduction

ENTERPRISE and home wireless local area networks (WLANs) are based on the CSMA/CA-based Distributed Coordination Function (DCF) medium access control (MAC) standardized in IEEE 802.11 [3] and commercially referred to as "WiFi." It is well known that, due to the intrinsic properties of the DCF MAC protocol, there are several limitations to the quality of service (QoS) offered by WiFi WLANs. For example, from Fig. 1, we see that when out of 3, 6, 9, 12, etc., stations (STAs), 1/3 of the STAs are associated with an access point (AP) at each of the rates 54, 24, and 6 Mb/s, then the aggregate bulk TCP-controlled download throughputs of the STAs associated at each rate are *the same* (just over 3 Mb/s: bottom three curves in Fig. 1), and the total download throughput is about 9.5 Mb/s (third curve from the top), whereas when all the STAs are associated at 54 Mb/s, then the aggregate throughput is about 22.5 Mb/s (the top curve); see also [4] for the same observation in the *saturated* node setting. In practical WLAN deployments, such mixed rate associations can be expected, thus leading to a reduction in the overall network throughput. For an analytical understanding of this phenomenon, in the context of TCP-controlled bulk transfers, see [5] and [6].

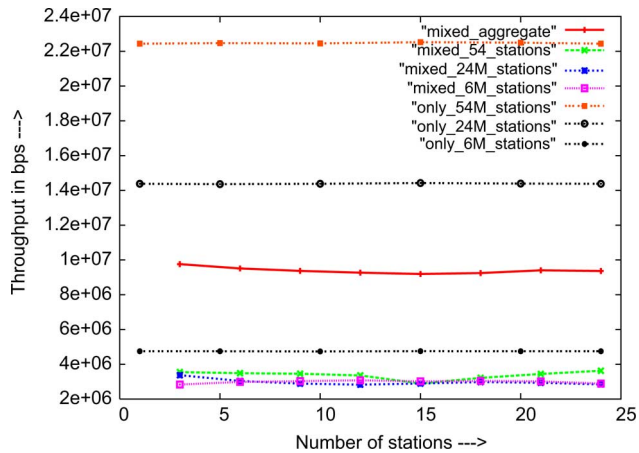

Fig. 1. Simulation results showing TCP download throughputs for n STAs when *all* n STAs are associated at the PHY rates of 54 Mb/s (plots labeled "only"), or at 24 Mb/s, or at 6 Mb/s, and also when 1/3 of the STAs are at each of the three rates (plots labeled "mixed").

Another QoS problem reported (see [7]) is that of unfairness between upload and download TCP-controlled bulk transfers for practical sizes of the AP buffer (which stores the packets contending for access to the wireless medium); see [8] for an analytical model that explains the observations in [7]. Real-time interactive traffic such as that due to Voice over IP (VoIP) telephony is known to perform poorly in the presence of TCP traffic in WLANs. To solve this problem, the IEEE 802.11e standard

Manuscript received July 31, 2011; revised January 31, 2012 and May 26, 2012; accepted June 07, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Bonald. Date of publication July 18, 2012; date of current version April 12, 2013. This work was supported by the Department of Information Technology (DIT), India, under a research grant and the Department of Science and Technology (DST), India, under a research grant.

M. Hegde, P. Kumar, K. R. Vasudev, N. N. Sowmya, S. V. R. Anand, and A. Kumar are with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India (e-mail: malati@ece.iisc.ernet.in; sowmya@ece.iisc.ernet.in; anand@ece.iisc.ernet.in; anurag@ece.iisc.ernet.in).

P. Kumar was with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India. He is now with Brocade Communications Systems, Bangalore 560103, India (e-mail: pava.k.pujari@gmail.com

K. R. Vasudev was with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India. He is now with the University of Kiel, Kiel 24118, Germany (e-mail: vasudevkr@gmail.com).

J. Kuri is with the Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore 560012, India (e-mail: kuri@cedt.iisc.ernet.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2207402

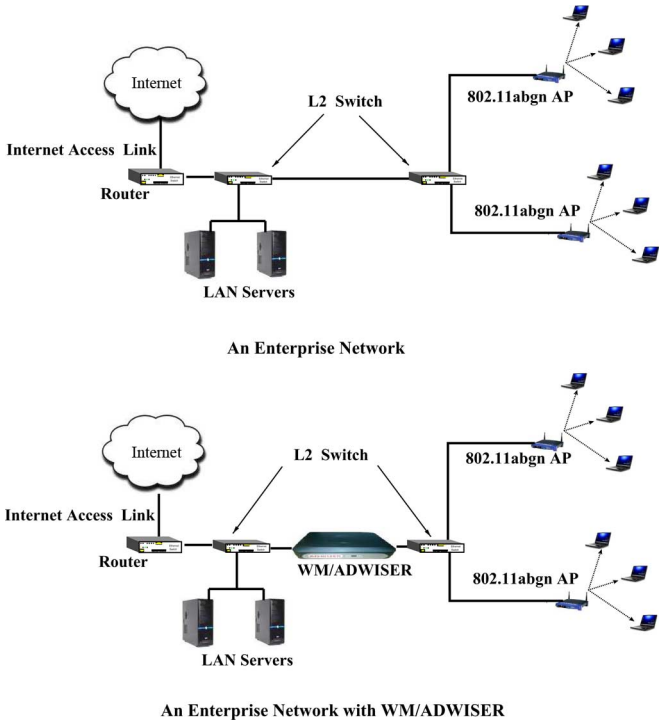

Fig. 2. *(top)* Enterprise network in which all traffic between the WLAN and the wireline local network passes through one Ethernet link. *(bottom)* Same network with WM or ADWISER in place.

provides an enhanced DCF (EDCF) in which STAs carrying voice transfers can obtain prioritized service. While these mechanisms serve to isolate VoIP traffic from TCP transfers, admission control of VoIP connections is needed, specially if some rate guarantees are to be provided to TCP transfers (see [9]).

We earlier proposed WLAN Manager (WM) [1], [2], a device through which all traffic between the WLAN and the wireline LAN passes (see Fig. 2). WM can be viewed as implementing a Split-MAC-based architecture (see the RFCs [10] and [11]) that works with off-the-shelf commercial products without requiring any changes in the APs. While retaining several design principles employed in WM [1], [2], in this paper, we extend the functionality to address the unfair Internet bandwidth sharing that arises when there is a mix of Internet–WLAN and LAN–WLAN TCP transfers. This unfairness can be explained as follows. The large difference in the round-trip times (RTTs) seen by these flows causes unequal buffer sharing at the AP in favor of flows with smaller RTTs, which results in the AP more often "serving" LAN–WLAN packets as compared to WAN–WLAN packets (see Section II-D for an experimental demonstration of this problem). We propose a cascaded scheduling mechanism that eliminates this unfairness problem. This additional new feature thus enables us to manage both the Internet access bandwidth and enhance WLAN performance of an enterprise having LAN and WLAN users, all within a single box, *without any modifications to the AP or the STAs*. We have appropriately named this evolution of WM as ADvanced Wi-fi Internet Service EnhanceR (ADWISER).

Contributions of This Paper: This paper is about implementation of ADWISER and its experimental evaluation. Our experimental work has been performed on a hybrid testbed (in which the PHY and MAC of the WLAN are modeled on the Qualnet simulator, whereas the servers, STAs, and the wireline LAN are physical entities) and also on a real testbed with a single AP and several STAs. We report our experiences in implementing ADWISER on an off-the-shelf Linux platform, and we provide the implementation details. We then report the highly encouraging experimental results obtained and the insights gained; we focus here exclusively on QoS management of TCP transfers.

In Section II, we provide an experimental review of the TCP QoS problems that ADWISER aims to solve. This review makes the paper self-contained and also helps to understand the results on the experimental evaluation of ADWISER. Furthermore, to the best of our knowledge, the experimental observations of starvation of the Internet access link during simultaneous wide area network (WAN)–WLAN and LAN–WLAN transfers, and the tradeoff between VoIP calls and TCP throughput, under IEEE 802.11e EDCF, are novel in the IEEE 802.11 WLAN setting. Section III describes the design principles behind ADWISER, some details of the system architecture, and the various queueing, scheduling, and service rate adaptation algorithms. In Section IV, we provide an extensive experimental study that illustrates the efficacy of ADWISER in solving the TCP QoS problems.

Related Literature: Since the publication of our proposal [1], another research group has independently reported a similar idea (see [12] and the references therein). While the general motivating concerns in the two proposals are similar, we have taken some specific TCP performance issues—namely: 1) fairness and efficiency due to different PHY rates; 2) fairness for arbitrary combinations of upload and download connections; 3) prevention of starvation of transfers from the Internet in the presence of transfers from the wired LAN; and 4) policy-based TCP throughput differentiation—to the stage of complete implementation as an ADWISER prototype. Since our implementation is not in the access point, in order to achieve these objectives, ADWISER has to adaptively learn the *effective service capacity* of the WLAN.

TCP-related QoS problems in WLANs have been addressed earlier by many authors. Multirate anomaly was addressed in [13], where time fairness was ensured using a Leaky Bucket. The work in [14] proposed the control of the "channel occupancy time" of a TCP connection by TCP window adjustments, based on link-layer measurements.

When STAs are associated with an AP at the *same* rate, fairness issues arise during simultaneous uploads and downloads. This was addressed in [15], where, to provide downloads adequate medium access opportunities, the value of the uplink TCP receiver's advertised window was reduced as the ACKs passed the AP. The proposal in [16] was to restrict the rate at which uplink data packets can reach the corresponding TCP receivers by incorporating a rate limiter inside the AP. On the other hand, the approach in [17] was to introduce separate queues for storing TCP data packets and ACKs and serving the latter with higher probability. The idea in [18] is similar, except that a weighted round robin (WRR) scheduler is used to serve the queued data and ACK packets. Yet another approach was followed in [19], where uplink packets were dropped deliberately at the AP to trigger congestion control mechanisms at the TCP sender. The

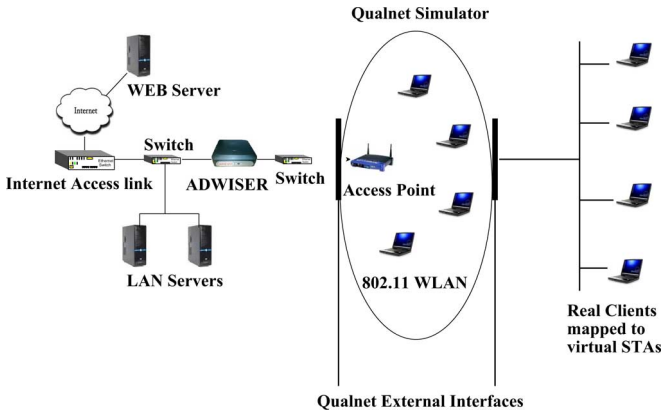

Fig. 3. ADWISER in a hybrid testbed consisting of real end-systems and a WLAN emulated in the Qualnet simulator.

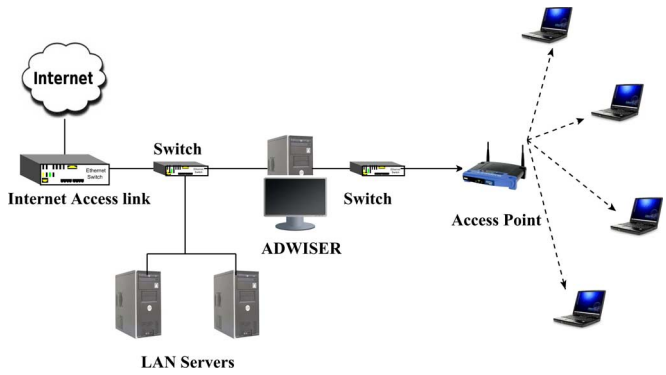

Fig. 4. Physical testbed for WAN-WLAN and LAN-WLAN transfers. By a software command, ADWISER can be bypassed or inserted.

approach in [20] is based on per-connection queueing and fair scheduling at the AP, with the purpose of releasing TCP ACKs in a controlled manner to slow down an uplink TCP connection's window growth. The Selective Packet Marking (SPM) scheme in [7] aims to protect the first few packets in a downlink transfer (crucial for window growth) by marking them as high-priority and requiring the AP to buffer them preferentially. Unfairness between simultaneous uplink transfers has been reported earlier in [21]. This paper provides one solution by invoking IEEE 802.11e service differentiation to handle downlink TCP ACKs in the AP; another solution is provided in [22], where the idea is to delay TCP ACKs by per-flow queueing.

Some authors have taken the approach of modifying MAC-layer characteristics to ensure improved fairness at higher layers. The proposal in [23] is to set the maximum transmission unit (MTU) of a station proportional to its rate of association; effectively, this ensures time fairness. In [24], the same effect is achieved by setting the initial congestion window of a station $\mathrm{CW}_{\min}$ to a value inversely proportional to the station's rate of association. The uplink-downlink fairness issue is addressed in [25] by utilizing the shortest interframe spacing PIFS for downlink access by APs. The work reported in [26] explores the use of the parameters $\mathrm{CW}_{\min}$ as well AIFS of the 802.11e standard to alleviate unfairness between competing uplink TCP transfers.

Thus, these approaches require either changes to MAC parameters and/or modifications to the firmware running on the AP. On the other hand, ADWISER does not require any modification to the MAC parameters or the AP firmware and works with any existing IEEE 802.11-based infrastructure WLAN.

II. TCP QoS Problems: Experimental Illustrations

In the process of understanding the QoS problems and developing ADWISER, we have used a *hybrid testbed,* that combines physical network devices and a QualNet simulator [27]. In our experimental evaluation section, we also report results from a testbed with all physical network devices.

The hybrid testbed is shown in Fig. 3 and permits us to experiment with a larger WLAN, in a controlled environment, than is practical to set up in our laboratory. The QualNet simulator (ver. 4.5) is used to emulate the MAC and PHY of the WLAN devices. QualNet provides a mechanism to inject actual packet traffic into the simulated wireless nodes through its external interface APIs. As shown in the figure, the hybrid testbed comprises a physical server, an Internet link emulator (8 Mb/s, 300 ms RTT emulated using Linux *netem*), a router, an Ethernet LAN switch, the Linux machine on which ADWISER is implemented, and laptops on which the client applications run. The QualNet simulator models the AP, the STAs (each corresponding to one of the laptops), and the wireless medium. Although Qualnet implements a rate-adaptation algorithm, this is turned off in our experiments, so that each STA is always associated at a fixed PHY rate with an AP. This facilitates comparison of experimental results to those predicted from analysis. Our TCP transfers are initiated from the end-systems, then the packets of the TCP flows pass through the simulated MAC in the QualNet simulator, thus experiencing the WLAN contention and delays as they would have in a real network.

Our experiments with actual access points were conducted on the *physical testbed* shown in Fig. 4. The testbed comprised a single WiFi Multimedia Extensions (WME) compliant Cisco Aironet AP (1240 series) and standard laptops running Linux (kernel release 2.6.23, Fedora 8 distribution). In our experiments, the AP is configured to associate STAs at a single rate, thus effectively disabling the rate adaptation algorithm. An 8 Mb/s Internet access link with a propagation delay of 300 ms was configured using the Linux *netem* utility.

Where we report experiments with Web traffic, we have used scripts that generate *Web-like* traffic. A "Web page" is modeled as containing multiple objects composed of text and images. Every page downloaded has a main file plus a random number of auxiliary files; this random number takes values between 1 and 6. After each page download, there is an inactive think time. The files in the Web page are sampled from a Pareto distribution [the complementary cumulative distribution function (cdf) has tail exponent $= 1.1$] with an average file size of 20 kB. On an average, every page has four files with an average size of 20 kB, and these files are downloaded in parallel over separate TCP connections from the Web server.

A. Multirate Unfairness

This problem was already mentioned in the Introduction and illustrated via Fig. 1, which was obtained from the hybrid testbed. The mixed PHY rate scenario leads to a drastic

reduction in the overall WLAN throughput, while equalizing connection throughputs, since the DCF MAC is *packet fair* and hence gives an equal chance for all backlogged nodes to contend, irrespective of the PHY rates at which they can send.

B. Upload-Download Unfairness

In addition to the literature discussed earlier, we have made observations about TCP upload-download unfairness that we briefly present here. Unfairness between competing TCP uploads and downloads has been addressed in [28] as well. However, apart from the forward/reverse path asymmetry mentioned in [28], the issue of AP buffer space is important, as the discussion below shows.

Let us first recall the reason for the upload-download unfairness reported in [7]. An analytical model for the observations in [7] can be found in [8]. In an infrastructure WLAN, all traffic between the STAs and a server on the wireline network must pass through the AP. We recall that the AP is one device that contends to send packets for all the connections, whereas each STA contends only for its own connection. The AP is thus a bottleneck in the infrastructure WLAN setting (see [5] and [6]). The packets from upload and download TCP connections (respectively, TCP ACKs and TCP data packets) fill up the AP queue, which is stored in a finite buffer. When the TCP windows grow, packet drops take place at the tail of the AP buffer. For the downlink transfers, data packet drops at the AP buffer affect the growth of the TCP congestion windows. Since ACK packets (corresponding to uplink transfers) are smaller, they suffer a smaller drop rate. Furthermore, TCP ACKs are cumulative, i.e., each ACK acknowledges all the packets up to the packet that it acknowledges. Hence, uplink transfers are not as severely affected by tail-drop packet losses at the AP buffer. In our experiments, we have found that with the recent versions of TCP that implement window scaling and the availability of large TCP receive buffers (8 MB was the default in our experiments), this reported upload-download unfairness can be very severe and is not relieved by increasing the AP buffer space; see the results reported later in this section.

To illustrate the unfairness problem between uplink and downlink traffic flows, we ran experiments on the hybrid testbed. Ten wireless STAs are connected at the 11 Mb/s PHY rate to an AP. STAs 1–5 each upload a large file to a local server (on the wired LAN) via the AP, while STAs 6–10 download a file of the same size from the server. All the transfers are initiated simultaneously using the Linux *wget* file transfer utility. The TCP receivers employ the delayed ACK mechanism. The buffer size in the AP is varied from 10 to 350 packets (15–525 kB buffer at AP), and the TCP maximum segment size (MSS) is set to 1500 B. The 10 transfers are allowed to run for a period of 300 s. Fig. 5 shows the aggregate upload and download throughputs for the two groups of five STAs each, for varying AP buffer size. We see that the five STAs uploading files to the server obtain by far the larger part of the network throughput. We observe that an increase in the AP buffer size has little impact on the aggregate throughput seen by both uplink and downlink transfers.

Figs. 6 and 7 show the evolution of the TCP congestion windows for the uplink and downlink transfers of these 10 STAs for an input AP buffer size of 100 packets (150 kB)[1]. Each DATA packet is 1500 B, whereas a TCP ACK packet is 40 B, plus fields for options such as timestamp and SACK (if any). Thus, the AP buffer can accommodate about 100 DATA packets, but 1000 s of TCP ACK packets. The round-trip time between the AP and the server on the LAN is negligible, hence, as explained earlier

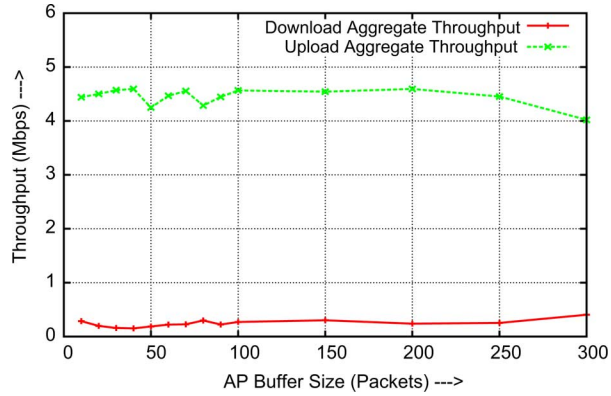

Fig. 5. Aggregate upload and download throughputs of 10 STAs (five performing uploads and five performing downloads) at 11 Mb/s with varying input AP buffer size. Hybrid testbed; TCP window scaling is enabled.

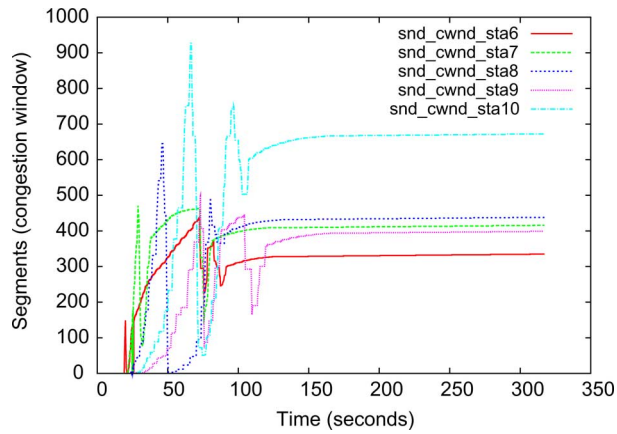

Fig. 6. Congestion windows (versus time) for the five upload TCP transfers, when five upload transfers and five download transfers are in progress. Hybrid testbed; AP buffer size $= 100$ packets.

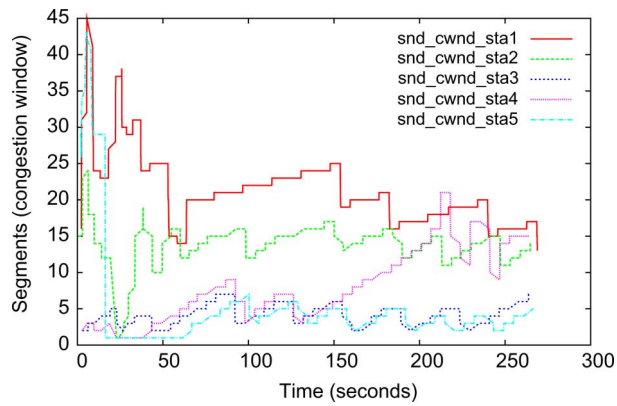

Fig. 7. Congestion windows (versus time) for the five download TCP transfers, when five upload transfers and five download transfers are in progress. Hybrid testbed; AP buffer size $= 100$ packets.

[1]Although the buffer size is specified in number of packets, it is maintained internally as an equivalent number of contiguous bytes.

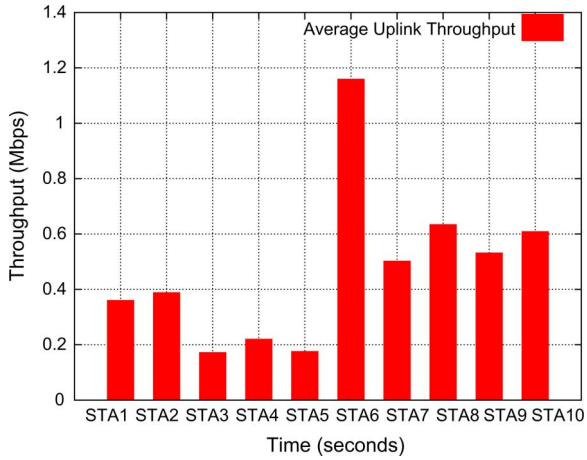

Fig. 8. Upload throughputs; 10 STAs associated at 11 Mb/s; hybrid testbed.

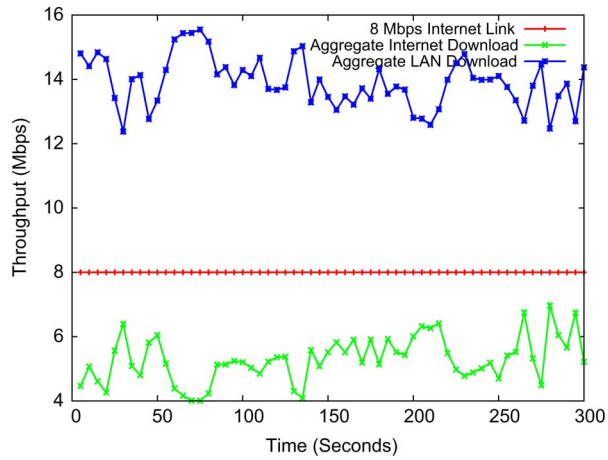

Fig. 9. Aggregate throughputs for WAN–WLAN and LAN–WLAN transfers; physical testbed of Fig. 4.

in this section, the congestion windows of all connections, reside almost entirely in the AP buffer, and the windows of the uplink transfers are able to grow to large values (300–700 segments), with the TCP ACKs then occupying more than 50% of the AP buffer (see Fig. 6). On the other hand, the congestion windows of the downlink transfers grow to just 5–15 segments, and the remaining AP buffer is occupied by these packets (see Fig. 7). Hence, the AP more often serves ACKs for the uplink connections, which therefore obtain a very high throughput as compared to the downlink transfers.

C. Unfairness Among Multiple Uploads

When there are only several upload transfers, a newly initiated TCP flow can lose ACK packets associated with its first few data transmissions, thus inducing a timeout. The ACK packets for the retransmitted data packets can also be lost, leading to further timeouts (with associated doubling of the retransmit timer), and so the flow can be completely starved for long periods. We conducted an experiment on the hybrid testbed with 10 STAs, all associated at 11 Mb/s rate with one AP. Upload transfers are initiated simultaneously from all the STAs and throughput observations are made for 300 s. The bar plot in Fig. 8 shows the average throughputs of the 10 upload connections. Only some of the STAs (6–10) obtain considerable bulk transfer throughputs, whereas some STAs (3–5) obtain much smaller throughputs. Thus, even among upload transfers, there can be random unfairness depending on loss events and the consequent unfairness in TCP window evolution.

D. WAN-WLAN and LAN-WLAN Unfairness

Unfairness among competing TCP flows at a bottleneck router with different RTTs is common in wired networks. The large difference in the RTTs causes unequal sharing of the bottleneck link, favoring the TCP connections with smaller RTT. The same phenomenon can be observed in the infrastructure WLAN network with STAs initiating transfers from Internet servers and local LAN servers. This causes unfair wireless channel access times between the competing flows from LAN-originated traffic and the Internet traffic destined to the WLAN (see also [29]). This unfairness can be explained as follows.

Consider the situation when both the transfers are downlink. The WAN–WLAN connection has a large round-trip propagation delay, and hence requires a large number of packets in flight in order to sustain its throughput. Losses at the AP buffer result in its window being reduced, much of the window is "in flight," and little remains in the AP buffer. On the other hand the LAN–WLAN connection requires a very small number of packets in flight, can rapidly recover from losses, and ends up occupying most of the AP buffer. Thus, since the AP renders first-in–first-out (FIFO) service to packets in its buffer, it finds a LAN–WLAN packet to serve with a much higher probability than that of a WAN–WLAN connection. This results in the throughput unfairness that we have found below in our experiments.

We conducted an experiment on the physical testbed shown in Fig. 4 with four STAs all associated at 54 Mb/s. TCP transfers were initiated on the two STAs downloading files from the Internet server. Simultaneously, from the other two STAs, TCP download transfers from the local LAN servers were started. Fig. 9 shows the aggregate TCP throughputs for Internet and LAN transfers. It is observed in Fig. 9 that the LAN–WLAN transfers see higher throughputs, and the Internet access link gets starved. This is undesirable since the Internet access link is often an expensive resource for a small enterprise.

As a consequence of the above unfairness, we observed that when we introduced Web-like transfers from the Internet server, the Web response times (the time taken to download a complete page from the Web server) increased substantially in the presence of LAN transfers.

The testbed shown in Fig. 4 was used for experiments with Web-like transfers. One station is performing Web-like downloads from the Internet server. In Fig. 10, from 100 to 300 s, there are only Web-like downloads, and during this period the average response time is 1.76 s for an RTT of 300 ms. Even though the Internet access link is 8 Mb/s, this response time is mainly dictated by the RTT, as the TCP connection setup followed by a Web download of a 20-kB file will take approximately five RTTs. Then, from 300 to 600 s, we initiated a TCP bulk download from the LAN server along with the Web-like downlink transfers. We observe (again, from Fig. 10) that the

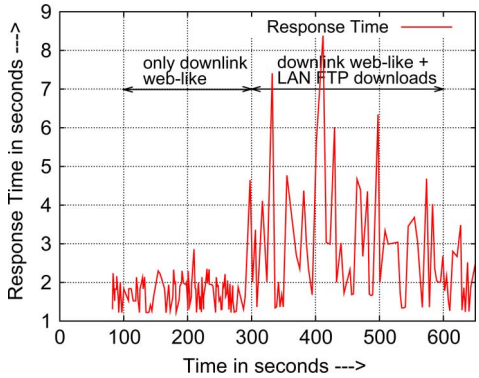

Fig. 10. Response times of Web-like downloads from the server on the WAN, in the presence of bulk download from the wireline LAN; physical testbed. The completion times of successive downloads are plotted as points joined by lines.

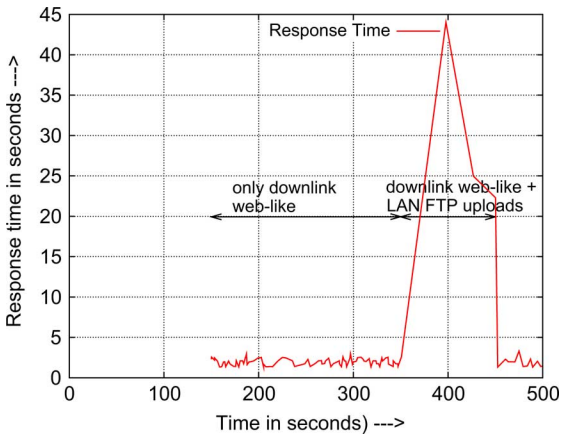

Fig. 11. Response times of Web-like downloads from the server on the WAN, in the presence of bulk uploads to the wireline LAN; physical testbed. The completion times of successive downloads are plotted as points joined by lines.

response times shot up to an average value of 2.60 s and became highly variable with large peaks.

We used the physical testbed to demonstrate the response time variation for Web-like downlink transfers in the presence of a bulk upload to a LAN server. In Fig. 11, from 150 to 350 s, one station downloads Web-like traffic from an Internet server. Here, we observe the average response time to be 1.85 s. From 350 until 450 s, we initiated a TCP bulk upload to the LAN server along with the Web-like downloads. We observe that the response time shot up to an average value of 30.43 s. This large increase of response time of Web transfers in the presence of a LAN upload as compared to a LAN download is due to the upload-download unfairness (Section II-B), which is exacerbated by the unfairness under discussion in this section.

E. Effect of VoIP Calls on TCP Throughput

It is well known that unless there is service differentiation between packet voice and TCP transfers, VoIP traffic obtains very poor performance. When service differentiation is provided, however, connection admission control (CAC) may need to be exercised for VoIP calls so as to guarantee some aggregate throughput for TCP transfers.

In IEEE 802.11 WLANs, service differentiation between voice and TCP packets is implemented by the EDCF mechanisms (standardized in IEEE 802.11e) that permit the voice access category to use more aggressive backoff and reattempt

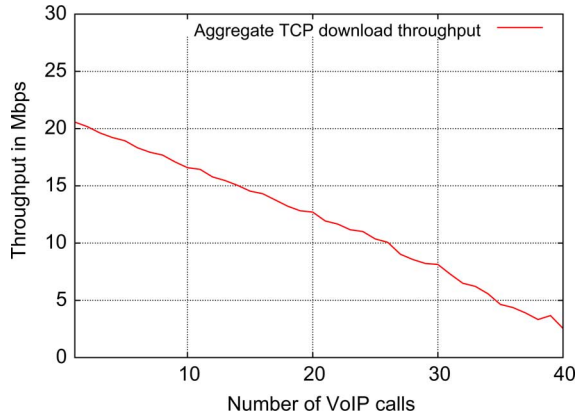

Fig. 12. Download throughput obtained by a single TCP download with increasing number of VoIP calls. Results are from the hybrid testbed. The MAC protocol is IEEE 802.11e, over 54-Mb/s PHY; VoIP: G.711 coding, 200-B packets; TCP DATA packets: 1500 B.

parameters. These mechanisms provide the voice traffic with an approximation to preemptive and rate priority. Harsha *et al.* [9] have provided an analytical model for the decrease in TCP throughput as the number of VoIP calls is increased. It was shown that as the number of voice calls is increased up to the voice call capacity, the TCP throughput drops linearly with the number of accepted voice calls. We have corroborated the reported behavior on our hybrid testbed shown in Fig. 3. In Fig. 12, we have plotted the TCP throughput versus the number of VoIP calls for 802.11g/e, with the VoIP traffic being carried on AC3. The STAs are associated at the PHY rate of 54 Mb/s, the VoIP codec is G.711 with a packet size of 200 B, and TCP data packet size is 1500 B. We have considered a single TCP download. As seen in Fig. 12, the TCP throughput is close to 2.5 Mb/s when the number of admitted VoIP calls reaches 40 (up to which point the target maximum delay of 20 ms for voice packets is respected). It can be seen that the TCP throughput drops roughly linearly at the rate of 0.45 Mb/s for each admitted voice call. Thus, from Fig. 12, we see that limiting the number of VoIP calls (of the above type) to, say, 10 guarantees that the TCP transfers are guaranteed an aggregate throughput of about 17 Mb/s.

III. ADWISER: Approach, Algorithms, and Implementation

A. ADWISER Approach [1], [2]

We have implemented ADWISER on a standard personal computer platform on Linux. The device is located (see Fig. 2) so that all packets that pass through the AP also pass through the device. Here, we present a conceptual overview of the ADWISER approach. Details are provided in Sections III-B and III-C.

In this paper, we assume that the user devices are all on the WLAN. These devices could have TCP connections to servers on the wireline LAN or on the Internet (see Figs. 3 and 4).

The basic ideas implemented in ADWISER [1], [2] are the following (see Fig. 13).

1) Virtual Servers and Queueing: If packets are allowed to accumulate in the AP and the STAs, then the usual behavior of IEEE 802.11 DCF shows up; see Fig. 1. ADWISER works

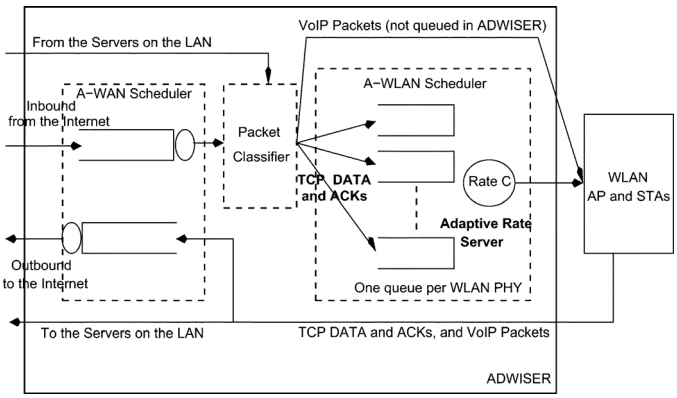

Fig. 13. ADWISER: Schematic of the scheduler that controls the sharing of the WLAN medium. All queues shown are hierarchical queues, so that service differentiation within each queue can be configured by the user.

by attempting to draw into itself the queues from the two bottleneck resources that it manages: the WLAN medium and the Internet access link. This is achieved by creating virtual servers inside ADWISER. Corresponding to the WLAN medium, there is one virtual server (with its associated queues) since the IEEE 802.11 WLAN presents itself as a half-duplex wireless link; we call this the ADWISER WLAN (A-WLAN) scheduler. On the other hand, two virtual servers (with their associated queues) correspond to the inbound and outbound directions of the full-duplex Internet WAN access link; we call this the ADWISER WAN (A-WAN) scheduler. Queueing is forced to take place in ADWISER rather than at the devices physically connected to the bottleneck links (i.e., the AP in the case of the WLAN medium, and the access routers in the case of the Internet access link) by setting the service rates of the virtual servers a little *less* than the effective service rates of the corresponding physical links. This approach also prevents packet drops from occurring due to buffer overflow in the AP. Ideally, with ADWISER, the queue buildup at the APs should be minimal, without starving the wireless medium. All queues shown in Fig. 13 are hierarchical in order to facilitate additional user-configured service differentiation. Coordination between the A-WAN and the A-WLAN schedulers is crucial for addressing the throughput unfairness reported in Section II-D and is described in Section III-C.2.

2) Flow of and Scheduling of TCP and VoIP Traffic: For TCP-controlled traffic, all packets flowing toward the WLAN (TCP DATA or TCP ACKs arriving from servers on the Internet or on the wired LAN) are classified and queued into per rate class queues behind the A-WLAN server.[2] These queues are then served by a virtual server using a fair queueing algorithm, which can enforce user-configured rate differentiation between the TCP transfers (see Fig. 13). Control of TCP download traffic (from the servers to the STAs) and of TCP upload traffic (from the STAs to the servers) is achieved by serving the TCP DATA packets and the TCP ACKs appropriately (see Section III-C.3). VoIP packets flowing toward the WLAN are placed in a strict priority queue in the A-WLAN server and essentially pass through freely into the AP. All packets flowing out of the WLAN toward the servers on the wired LAN are passed through ADWISER without any further queueing. All packets arriving into ADWISER from the WAN access link are queued in the A-WAN scheduler at the virtual server corresponding to the inbound direction of the Internet access link. Similarly, packets arriving from the WLAN into ADWISER and destined for the Internet access link are queued in the A-WAN scheduler behind the virtual server corresponding to the outbound direction of the Internet access link.

3) Rates of the Virtual Servers: Whereas there is a well-defined bit rate of the Internet access link, e.g., 8 Mb/s in our experiments, the effective service rate of the WLAN medium depends on the medium access overheads and the channel conditions prevailing on the wireless links between the AP and the STAs. Furthermore, since IEEE 802.11e service differentiation is used between VoIP connections and TCP connections, the effective service rate that the WLAN medium provides to the TCP connections depends on the number of (and the coding used by) the VoIP connections. A key requirement in ADWISER is that the rate of the virtual server in the A-WLAN scheduler is dynamically adjusted by an online rate adaptation algorithm. This service rate adaptation is crucial to the proper working of ADWISER. If the service rate is too high, packet accumulation occurs in the AP and not in ADWISER, and thus ADWISER loses the ability to enforce the desired service ratio between the packets of the various connections. On the other hand, if the A-WLAN server's rate is too small, then the WLAN starves, and the system is inefficient. We have implemented a simple service rate adaptation algorithm that dynamically adapts the service rate, even as the number of STAs associated at each rate varies; see Section III-C.1.c for details.

4) VoIP CAC: ADWISER can perform VoIP connection admission control in order to provide desired aggregate throughput to the TCP transfers. Having admitted a VoIP call, ADWISER exercises no additional control on VoIP packets, letting the IEEE 802.11e mechanisms take care of providing the necessary service differentiation between VoIP and TCP transfers.

B. ADWISER Software Architecture

A schematic of the high-level ADWISER software architecture is depicted in Fig. 14. In the *packet capture module*, we capture the incoming traffic from the wired link toward the AP. The idea is to buffer only the packets coming from the wired link, going toward the WLAN, in order to achieve the desired fairness in both directions. Note that this is possible for TCP upload connections since the flow of ACKs in the downlink direction can be controlled, thereby controlling the TCP throughput. The *packet classifier module* places an arriving packet into one of the physical queues. The information in the IP and TCP headers is used to form a 4-tuple (source address, destination address, source port, and destination port) in identifying the direction and the leaf node in the queueing hierarchy. The voice packets are given strict priority and appropriate

[2]Even in the presence of rate adaptation, we have found experimentally (with the Minstrel algorithm; see [30]) that, for various ranges of received signal strength indicator (RSSI) values, there are dominant rates at which packets are sent between the AP and an associated STA. SNMP polls to the AP elicit the RSSI values at which packets are received from an STA. These RSSI values can be used to map the STAs into, say, three or four rate classes. In our experiments, however, rate adaptation was effectively disabled: in the hybrid experiments by turning rate adaptation off in the Qualnet simulator, and in the physical AP experiments by permitting the AP to associate STAs only at a particular rate, e.g., 54 or 11 Mb/s.

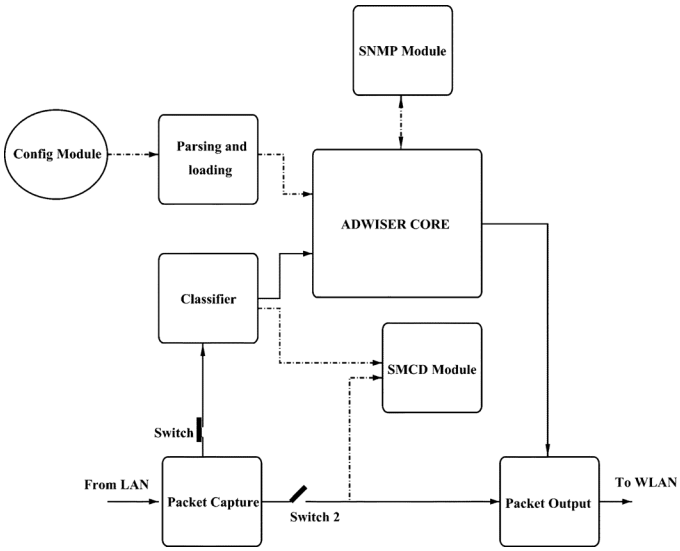

Fig. 14. ADWISER: High-level software architecture. When the positions of the two "switches" are as shown, then ADWISER is in the path of the traffic flow; when both positions are reversed, ADWISER is bypassed.

DSCP marking is done in ADWISER before forwarding to the AP. These marked packets get mapped to the voice access category, AC3, within an 802.11e AP. ADWISER therefore takes advantage of the QoS offered by 802.11e. The *SNMP module* periodically polls the APs to obtain information on their associated STAs; this relationship is explained later in Section III-C.1.c. The *SMCD module* (where SMCD expands to Statistics and Measurements Capture Daemon) captures and stores various statistics of the traffic passing through ADWISER.

C. Algorithms in ADWISER Core

The ADWISER Core (see Fig. 13) is where all the QoS algorithms are implemented. In this section, we describe these algorithms in detail.

1) Service Rate in the A-WLAN Scheduler:

a) Effective Service Rate for a PHY: ADWISER uses SNMP polls to the AP to determine which STAs are associated and the PHY rates at which they are associated. Then, for an STA associated at PHY rate, say R_i, the value of C_i, the effective bit rate at which TCP-controlled transfers take place between the AP and this STA, if this was the only connection, is calculated as follows:

$$C_i = \frac{(d \times L_{\mathrm{DATA}}) + L_{\mathrm{ACK}}}{(d \times (\frac{L_{\mathrm{DATA}}}{R_i} + T_{\mathrm{OVH}})) + (\frac{L_{\mathrm{ACK}}}{R_i} + T_{\mathrm{OVH}})}$$

where $d = 2$ to take care of delayed TCP ACKs, and T_{OVH} is the overhead time due to the medium access (RTS/CTS for TCP DATA and Basic Access for TCP ACK), the various interframe spaces, and the MAC ACK. Since ADWISER's attempt is to draw the queues into itself, thereby minimizing contention on the WLAN, when computing T_{OVH}, just one back-off duration is taken, and it is assumed that collisions do not take place. As will be seen below, any errors in the computation of the C_i values only affect the weights ϕ_i that are used in the A-WLAN scheduler.

b) Need for Service Rate Adaptation: The methodology to achieve the desired throughput fairness requires the ADWISER schedulers to serve the packets at certain desirable service rates; recall the discussion in Section III-A. We need to obtain the A-WLAN scheduler service rate that permits the queueing to take place in ADWISER so that it has control over the TCP throughputs, while keeping the WLAN maximally utilized.

Here is how we can think of the service rate. Suppose there are m STAs associated with the AP, at the PHY rates $r_i, 1 \leq i \leq m$. Each is carrying out a large file download from a server on the wired LAN. Let us imagine m queues in ADWISER, one for each station. The queues contain all the application-level bits that need to be transmitted on the wireless medium for that connection. *Thus, we imagine that, for each downlink data packet, even its uplink TCP ACK is queued in ADWISER; for each downlink ACK packet, the corresponding TCP data packets that will be generated are also queued in ADWISER.*

Suppose that ideal (bit-level) fair queueing is used to serve these queues using the weights $\phi_i = (\phi_1, \phi_2, \ldots, \phi_m)$, with $\sum_{j=1}^{m} \phi_j = 1$. Suppose all the queues are backlogged. Then, out of b bits (where, b is a large number) sent by ADWISER, $\phi_i b$ bits belong to connection i. With C_i as defined earlier in this section, the time occupied on the medium by these bits from connection i is $\frac{\phi_i b}{C_i}$. The total time taken to transmit all the b bits from the m connections is then given by $\sum_{i=1}^{m} \frac{\phi_i b}{C_i}$. Dividing b by this expression yields the effective rate at which the medium will carry bits above the MAC layer, i.e.,

$$C^* = \frac{1}{\frac{\phi_1}{C_1} + \frac{\phi_2}{C_2} + \cdots + \frac{\phi_m}{C_m}}. \tag{1}$$

Thus, C^* depends on the weights, $\phi_i, 1 \leq i \leq m$, and the PHY rates at which the STAs are connected.

Now, the idea of serving the queues of actual packets in ADWISER is the following.

1) Virtually replace each packet in the ADWISER queue by the number of higher layer bits (above MAC and PHY) that will need to be sent on the medium if the packet is released into the WLAN (details in Section III-C-III).
2) Adapt the service rate of these queues (of virtual bits) so that the rate is a little less than C^*. Adaptation of C^* is needed since the population of active STAs, and the PHY rates at which they are associated will keep varying over time. In addition, the C_i are affected by VoIP calls.

As an example, if m_j STAs are associated at rate $r_j, 1 \leq j \leq n$, then we have n queues in ADWISER. Consider the following weights, for $1 \leq i \leq n$:

$$\phi_i = \frac{m_i C_i}{\sum_{j=1}^{n} m_j C_j} \tag{2}$$

i.e., *proportional fairness* or *time fairness* (i.e., the target throughputs are proportional to the PHY rates of the connections), then

$$C^* = \frac{\sum_{i=1}^{n} m_i C_i}{\sum_{j=1}^{n} m_j}. \tag{3}$$

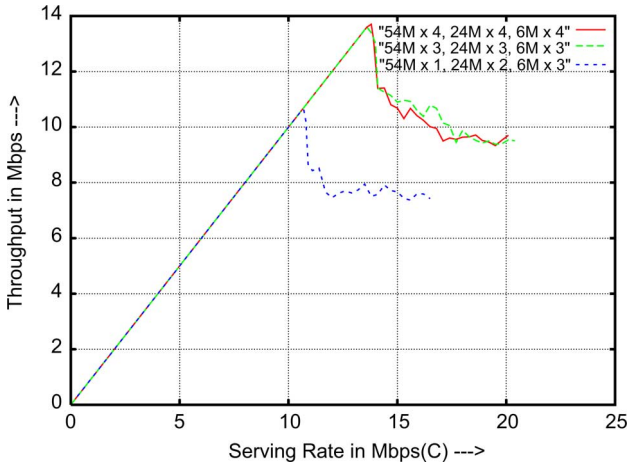

Fig. 15. Aggregate throughput for bulk file downloads, with ADWISER, for various numbers of STAs associated at 54, 24, and 6 Mb/s (IEEE 802.11a) plotted versus the ADWISER service rate C; results from the hybrid testbed.

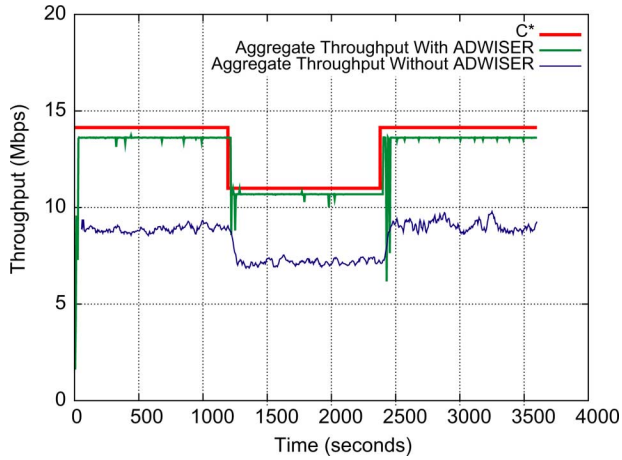

Fig. 16. ADWISER service rate adaptation: aggregate download throughput with and without ADWISER when the number of STAs associated at each rate varies over time; results from the hybrid testbed. C^*: thick line. With ADWISER: medium line. Without ADWISER: thin line.

The above discussion is illustrated by the hybrid experiment (recall Fig. 3) results shown in Fig. 15, where the scenario is that m_1, m_2, and m_3 STAs are associated, respectively, at the PHY rates of 54, 24, and 6 Mb/s. In ADWISER, there is one queue for each rate class. The service weights are given by the expression in (2). The ADWISER service rate C is then varied, and the aggregate download throughput is plotted versus C. There are three plots, corresponding to $m_1 = m_2 = m_3 = 4$, $m_1 = m_2 = m_3 = 3$, and $m_1 = 1, m_2 = 2, m_3 = 3$. The calculated values of C^*, from (3) (where the various parameter values from IEEE 802.11a are used), are 14.14, 14.14, and 11.00 Mb/s. We notice that as C increases from a small value, the aggregate throughput is equal to C, showing that ADWISER is the bottleneck and the wireless medium is underutilized. However, when C approaches C^*, the aggregate throughput drops sharply and stabilizes at a value significantly smaller than C^*, as C further increases. Basically, the queues move into the AP and the default behavior takes over, i.e., in the mixed PHY rate situation, the aggregate throughput drops due to the prevalence of packet fairness rather than time fairness.

c) Service Rate Adaptation Algorithm: Above, we arrived at the service rate C^* analytically for a set of weights $\phi_i, 1 \leq i \leq m$. The experimental result in Fig. 15 illustrates the necessity of adapting the ADWISER service rate as the number of STAs associated at each rate changes over time. There is another important reason why the ADWISER service rate needs to be adaptively learned and cannot simply be computed from (1). Note that the rates C_i may be achieved only if there are no wireless channel losses. When there are losses, due to the signal-to-interference-plus-noise ratio (SINR) for connection i not being the best possible for the PHY rate at which the STA is associated, then more packets will be sent on the medium (due to MAC level retransmissions) than are accounted for by introducing virtual bits in the ADWISER queues. These losses will result in a smaller value of C_i, which will show up if the ADWISER service rate is adapted based on online measurements.

Based on the insights gained from Fig. 15 we have implemented the following simple adaptation algorithm. The service rate C of the A-WLAN scheduler in ADWISER is iteratively obtained as follows.

1) Initialize C to a value less than C^*.
2) Carry out the following update after each measurement interval (taken to be 1 s in our implementation).
   a) If the measured TCP throughput over the WLAN is equal to C, then
   $$C \leftarrow \min\{C^*, C + \Delta\}$$
   where Δ is a tunable parameter. We found 0.1 Mb/s to be good value for Δ.
   b) If the measured TCP throughput is less than C, then
   $$C \leftarrow C - \Delta.$$
3) In parallel with the above steps, the AP continues to be polled in order to obtain the most current information about STA association. If there is any change, then C^* is recomputed as above.

To illustrate the performance of our approach, we provide a sample result from the hybrid testbed; see Fig. 16. The setup is the same as that for the "open-loop" experiment conducted for Fig. 15. Now, the number of STAs in each rate class is changed at an interval of 20 min in the following order:

- three STAs in each rate class for first 20 min;
- one, two, and three STAs in 54-, 24-, and 6-Mb/s rate classes, respectively, for the next 20 min;
- four STAs in each rate class for the last 20 min.

In Fig. 16, we show the values of C^* for each interval and the service rate C to which ADWISER adapts. We also show the aggregate throughput without ADWISER; this is the bottom plot in the figure. We see that with ADWISER, the aggregate throughput tracks (but falls a little short of) the computed value of C^*. When C^* changes, the adaptation takes 50–60 s.

2) A-WAN and A-WLAN Scheduler Coordination: We have assumed that the Internet access link is an expensive resource and therefore would like it to be fully utilized whenever possible. Thus, WAN–WLAN transfers are allowed to obtain whatever aggregate throughout they can get. This aggregate throughput is measured and subtracted from C, the service rate of the A-WLAN scheduler, which is obtained as explained in

Section III-C.1.c. The LAN–WLAN transfers are then allowed to share the residual rate. The importance of setting C correctly is, thus, further underlined. If C is set too high, then the queueing will shift from ADWISER back to the AP, resulting in the phenomenon observed in Section II-D. On the other hand, if C is set too low, then the WLAN medium will be underutilized.

3) A-WLAN Packet Scheduler: ADWISER maintains separate queues for TCP DATA packets for download connections and TCP ACK packets for upload connections. The ADWISER packet scheduler releases these packets into the WLAN. A TCP ACK packet transmitted by the AP results in a certain number of uplink DATA packets toward the AP. One issue we need to address when trying to use an existing fair queueing scheduler, meant for a wired full-duplex link, is the half-duplex nature of the shared wireless link. The A-WLAN packet scheduler therefore has to release DATA and ACK packets in such a way that it accounts for the time required on the medium for the packets that will be triggered by the reception of these packets by the STAs.

When an ACK packet corresponding to the uplink traffic is served, in the TCP steady state, this results in two uplink DATA packets at the STA because of TCP's delayed ACK mechanism. The scheduler replaces each packet queued in the ADWISER buffer with a number of "virtual bits" corresponding to the number of bits that will actually be carried by the WLAN MAC and PHY as a result of the packet being released into the WLAN. Thus, for each ACK, the scheduler assumes a number of virtual bits equal to an ACK and two DATA packets. For a DATA packet, the scheduler assumes a number of virtual bits equal to a DATA packet and half an ACK (to roughly account for delayed ACKs). These are, of course, approximations; in practice, the ratio of ACKs and DATA packets can vary depending on the TCP state. In practice, we have found our simple approach to work quite well.

Packet scheduling is performed using the Start Time Fair Queuing (STFQ) [31] scheduling policy by suitably tagging start and finish numbers to each packet and scheduling the packet transmissions appropriately. Let us consider an arrival of a packet, $k+1$, of *virtual* length $l_{k+1}^{(j)}$ (see above) into a queue j at arrival instant $a_{k+1}^{(j)}$. Let $F_k^{(j)}$ denote the finish number of packets k in queue j. $V(t)$ denotes the (global) virtual time at time t. The start number $S_{k+1}^{(j)}$ is computed as

$$S_{k+1}^{(j)} = \max\{F_k^{(j)}, V(a_{k+1}^{(j)})\}. \tag{4}$$

Then

$$F_{k+1}^{(j)} = S_{k+1}^{(j)} + \frac{l_{k+1}^{(j)}}{\phi_j} \tag{5}$$

where $F_0^{(j)} = 0$. In STFQ, instead of computing the virtual time from a simulation of the corresponding GPS system, the following approximations are made. The virtual time is initialized to 0 and increases in jumps as follows. When a packet arrives (say, at t), and there is a packet in service, then STFQ approximates $V(t)$ as the start time of the packet in service. If the packet arrives, at time t to an idle system, then the value of $V(t)$ is taken to be the finish time of the last packet in the previous busy period. See [31] for details.

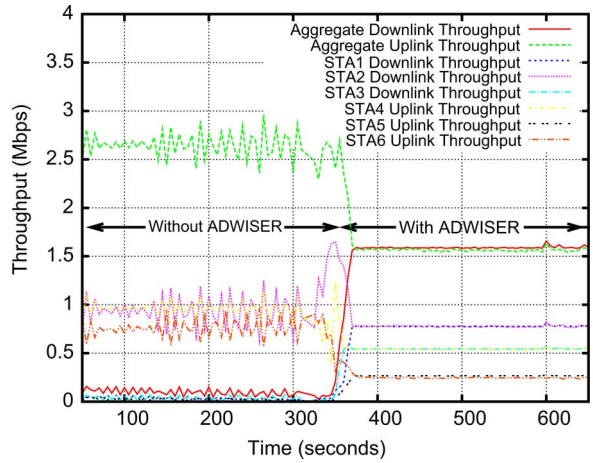

Fig. 17. Hybrid testbed: Uplink and downlink throughputs versus time for six STAs associated at 11-, 5.5-, and 2-Mb/s PHY rates, with and without ADWISER.

Packets are released into the WLAN in the order of their *start numbers*. This is because the actual server is the wireless medium itself. After releasing a packet into the WLAN from queue i, the ADWISER scheduler allows time for the virtual bits corresponding to the packet to be served at the current ADWISER service rate C.

IV. Experimental Results

A. Results From the Hybrid Testbed

Recall Fig. 3, which showed a schematic of the hybrid testbed. The wireless PHY and MAC are simulated in Qualnet. The WAN between the Web servers and the ADWISER is emulated by a computer running the Linux *netem* link emulator, in which a propagation delay of 300 ms is configured. The AP buffer is set to the default size of 100 packets. The wireless STAs and the Web server run on Linux. We have used the default Linux kernel settings. Thus, TCP window scaling is enabled, and delayed acknowledgment is on. Bulk file transfers (upload and downloads) are started between the STAs and the Web server. ADWISER is configured to provide equal aggregate throughputs to uploads and downloads. In the experiments described below, initially ADWISER is bypassed (cases labeled "Without ADWISER"), and later ADWISER is introduced (cases labeled "With ADWISER").

1) STAs Associated at Multiple PHY Rates: We consider six wireless STAs, with half of them downloading files from the Web server and the other half uploading files. The six wireless STAs are grouped into three sets of two STAs, each set associated at 11, 5.5, and 2 Mb/s, respectively. The plot in Fig. 17 shows the throughputs. During the initial 300 s, ADWISER has been bypassed. We notice that the download transfers obtain very small throughputs (the cluster of four plots near the bottom, including the solid plot for aggregate throughput). Recall the discussion in Section II-B. The upload transfers obtain equal throughputs of 0.8–0.9 Mb/s (the three jagged plots in the middle), with an aggregate throughput of about 2.6 Mb/s (topmost plot). After ADWISER is introduced, the upload and download throughputs for each PHY rate become equal; the aggregate upload and download throughputs become

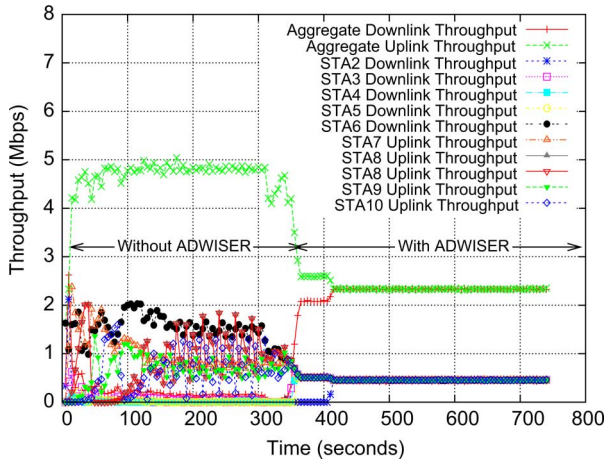

Fig. 18. Hybrid testbed: Throughputs of 10 STAs, five of which are performing uploads and five performing downloads. The STAs are associated at 11 Mb/s with an AP. Results with and without ADWISER are shown.

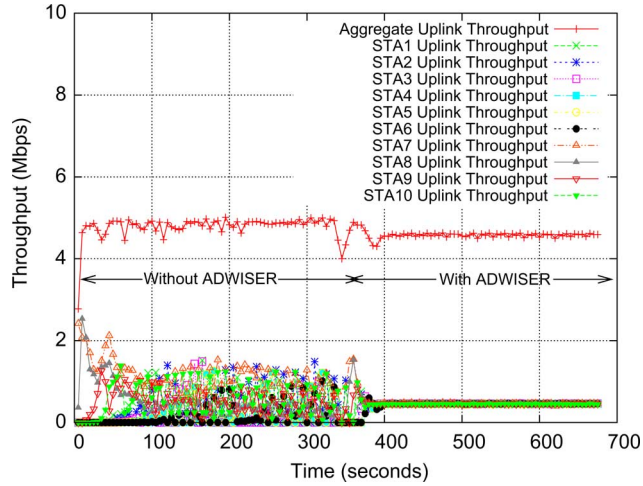

Fig. 19. Hybrid testbed: Upload throughputs obtained by 10 STAs associated at 11 Mb/s with and without ADWISER. The uppermost plot (+ points) shows the aggregate throughput.

about 1.578 Mb/s each, totaling to 3.156 Mb/s, which is just less than $C^* = 3.26$ Mb/s obtained from the parameters of this experiment. Furthermore, proportional service differentiation leads to the 11-, 5.5-, and 2-Mb/s transfers obtaining aggregate throughputs of, respectively, 2×0.778 Mb/s $= 1.556$ Mb/s, 2×0.545 Mb/s $= 1.09$ Mb/s, and 2×0.255 Mb/s $= 0.51$ Mb/s, which are roughly in the ratio of the $C_1 = 4.89$ Mb/s, $C_2 = 3.33$ Mb/s, and $C_3 = 1.58$ Mb/s, the ideal contention-free TCP throughputs at 11, 5.5, and 2 Mb/s. *Thus, ADWISER provides a higher aggregate throughput for the network, while providing upload-download fairness, and proportional throughput differentiation across the rate classes.*

Relation to the Literature: Time-fairness was achieved in [13] by a Leaky-Bucket-based algorithm, whereas the proposal in [14] was to control a station's channel occupancy time by a TCP window size adjustment algorithm, using channel occupancy measurements at the MAC layer. Both require modifications to the AP software. In contrast, our solution is completely transparent.

2) STAs Associated at a Single PHY Rate: In the setup shown in Fig. 3, 10 STAs are now associated with the AP at the 11 Mb/s PHY rate, keeping the rest of the setup the same. The following experiments are concerned with fairness between upload and download TCP transfers and between just upload TCP transfers.

Uploads and Downloads: In Fig. 18, we have plotted the throughputs of 10 TCP transfers one from each of 10 STAs, half of which perform uploads and half that perform downloads, of large files to and from the Web server. We observe from the "Without ADWISER" segment that the download throughputs are very small; these are the plots near the bottom, before 300 s. On the other hand, the aggregate upload throughput is about 4.8 Mb/s. ADWISER is introduced at 300 s, and after a transient period, the aggregate upload and download throughputs become equal (about 2.4 Mb/s each), and so do the individual throughputs of the 10 file transfers (about 0.48 Mb/s each). The transient period can be explained from the observations we had made in Section II-B. With ADWISER bypassed, 1000 s of ACKs of the upload connections accumulate in the AP buffer. It takes time to drain out these ACKs. Furthermore, ADWISER does not take control of the queues until the service rate C adapts. Of course, such a large transient is not expected to occur during normal operation with ADWISER in place.

Relation to the Literature: The key idea in [15] was to prevent uplink senders from sending too much data by presenting them with an artificially reduced receiver advertised window. The approach in [17] was to store data and ACK packets in two different queues, serving the latter with lower probability. Yet another proposal [19] was to trigger TCP congestion control actions by deliberately dropping uplink packets. "Virtual Flow Queueing" in [20] is based on adapting the well-known Weighted Fair Queueing (WFQ) policy to the wireless context. Packet lengths are inflated by adding "virtual bits" that account for MAC/PHY and interframe space overheads, as well as backoff intervals. The high-level approach outlined in [12] is also similar to ours.

Nearly all the proposals mentioned above are meant to be implemented on APs. Our transparent solution does not require per-connection TCP window manipulation, or TCP ACK policing using Leaky Buckets, and, unlike [20], uses STFQ scheduling to handle a link with varying service rate.

Uploads Only: There are 10 upload file transfers from 10 STAs. Fig. 19 shows the results from the hybrid testbed. Before ADWISER is introduced, there is considerable unfairness between the upload throughputs. Recall the discussion in Section II-C. On the other hand, with ADWISER, the upload throughputs become exactly equal.

Relation to the Literature: To address upload-upload unfairness, [22] uses a TCP ACK control mechanism at the AP that regulates the number of ACKs sent out according to the estimated average bandwidth each flow gets. The approach in [19] is to regulate TCP traffic by inducing TCP congestion through packet drops. Malone *et al.* [21] aim to solve the problem by invoking IEEE 802.11e service differentiation to prioritize TCP ACKs in the AP. We use a queueing-based approach for controlled release of TCP ACKs.

3) WAN-WLAN and LAN-WLAN Downloads: An 8 Mb/s Internet link between the Web servers and the gateway router, with a propagation delay of 300 ms, is emulated using the Linux *netem* utility. We consider a group of two STAs downloading

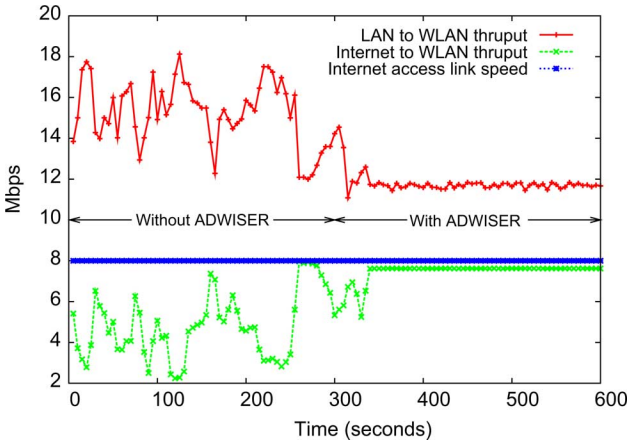

Fig. 20. Physical testbed: Aggregate TCP download throughputs for WAN–WLAN and LAN–WLAN transfers with and without ADWISER.

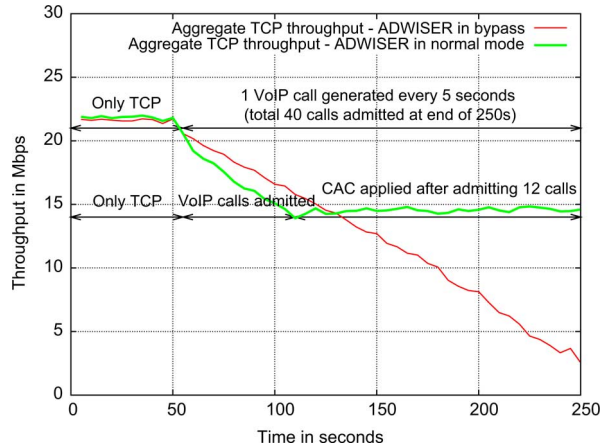

Fig. 21. Hybrid testBed: TCP throughput in the presence of VoIP calls with and without ADWISER. ADWISER bypassed: thin line. ADWISER in-line: thick line.

bulk files from the Internet Web server and another group of two STAs downloading bulk files from the LAN server. The plot in the Fig. 20 shows the bulk file throughputs "Without ADWISER" and "With ADWISER." We notice that without ADWISER, the Internet access link is poorly utilized in the presence of TCP transfers from the local LAN servers. In contrast, ADWISER eliminates the unfair wireless channel access times between the competing flows from LAN traffic and the Internet traffic destined to the WLAN. The Internet transfer rates are now extremely stable, with higher aggregate throughputs that are close to 8 Mb/s. The LAN–WLAN transfers see stable but reduced throughputs; their throughput is reduced since the WAN–WLAN transfers need to obtain more medium time on the WLAN.

Relation to the Literature: To the best of our knowledge, existing solutions do not address this problem.

4) Connection Admission Control for VoIP Calls: We have implemented VoIP CAC in ADWISER to provide aggregate throughput guarantees to TCP transfers in the presence of VoIP traffic.

Using the hybrid testbed in Fig. 3, we reported the effect of VoIP calls on TCP download throughput in Section II-E. To demonstrate the VoIP CAC feature in ADWISER, we had a single STA perform TCP bulk download from the LAN server, and then we started generating G.711 VoIP calls at the rate of one call every 5 s. In Fig. 21, up to 50 s, there is only a bulk TCP download from a server on the LAN to the STA. After 50 s, we started generating VoIP calls to other STAs. It is observed that, without ADWISER, all the VoIP calls are admitted, and the TCP throughput drops to as low as 3 Mb/s as the calls are added. When ADWISER is in place, with a CAC limit of 12 calls, it is observed that after admitting 12 calls (at about 110 s), no more VoIP calls are admitted. As a result, the TCP download throughput drops only to about 14 Mb/s and stays there.

Relation to the Literature: CAC for VoIP traffic is not new, but exercising it for providing QoS to TCP in the IEEE 802.11e context is a novel feature in ADWISER. We note here that, in practice, the CAC limit for VoIP calls will depend on their coding rates. Thus, ADWISER will need to have a matrix showing the combination of numbers of calls of each coding to be admitted. As an example, see [6] for an analytical approach for obtaining CAC limits for multiclass VoIP calls for IEEE 802.11 DCF.

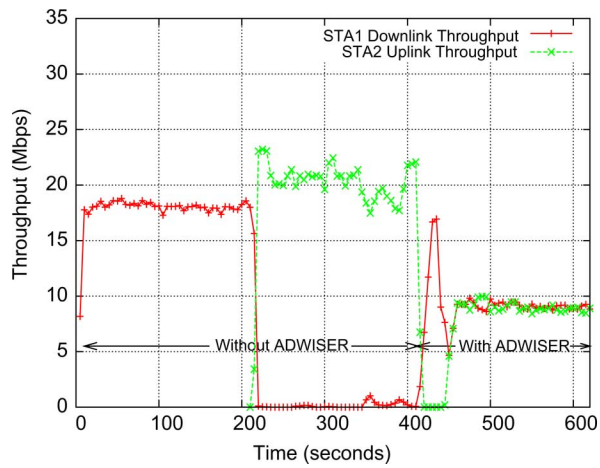

Fig. 22. Physical testbed: Upload and download throughputs for two STAs associated at 54 Mb/s (IEEE 802.11g) with and without ADWISER.

B. Results From a Testbed With a Real AP

In these experiments, we have an IEEE 802.11g Cisco Aironet AP with which two Linux-based laptops (STA1 and STA2) are associated, each at 54 Mb/s. The remaining experimental testbed and parameters are the same as in Section IV-A. Delayed ACKs were enabled in the TCP receivers; RTS/CTS was used to send data packets, whereas Basic Access was used for TCP ACKs. With this, the maximum possible throughput on the wireless medium (with 54 Mb/s PHY rate) can be calculated to be 23.14 Mb/s [9]. The actual throughputs achieved will, of course, be less.

Upload and Download: See Fig. 22. A large file download from the wireline Web server was initiated from STA1 and was allowed to run for 200 s; we observe a throughput of about 18.56 Mb/s. Then, an upload was started from STA2; the upload throughput was 21.2 Mb/s, whereas the download throughput fell to 0.34 Mb/s. The upload transfer obtains a slightly higher throughput than a download alone because some of the TCP

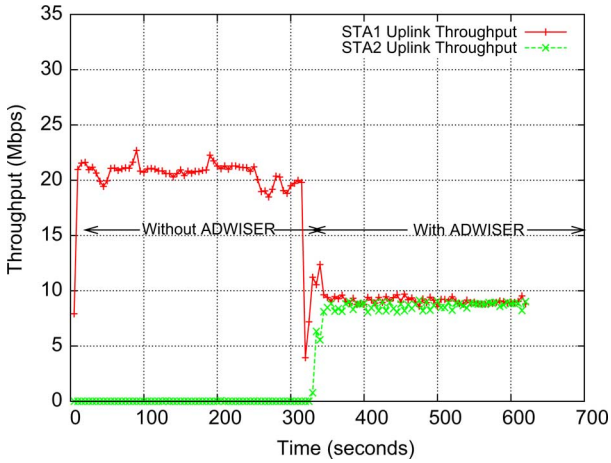

Fig. 23. Physical testbed: Upload throughputs for two STAs associated at 54 Mb/s (IEEE 802.11g) with and without ADWISER.

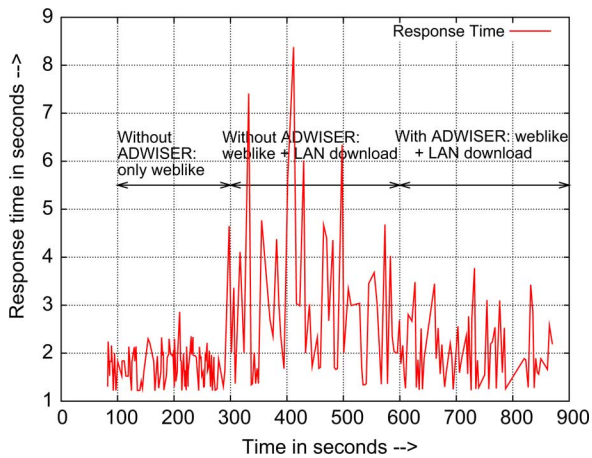

Fig. 24. Physical testbed: Response times of Web-like downloads in the presence of bulk download from the LAN, with and without ADWISER.

ACKs are lost in the AP, and hence fewer bits are carried on the medium per data packet. Once ADWISER is turned on at 400 s, the throughputs equalize, giving a total throughput of about 19 Mb/s. Recall, from Section III-C.1.c, that we adapt the ADWISER service rate to a value a little less than C^*, which is the reason for the lower aggregate throughput than what might ideally be expected.

Two Uploads: This experiment demonstrates unfairness among uploads and ADWISER's ability to enforce fairness. In Fig. 23, in the "Without ADWISER" period, only the upload from STA1 obtains throughput (about 21.5 Mb/s), whereas the one from STA2 is starved completely. This complete starvation of an STA, when performing TCP uploads, was also reported by Malone *et al.* [21]. Once ADWISER is introduced, both the uploads obtain equal throughputs, with the aggregate being 19 Mb/s.

Web-Like Traffic: The following two experiments were carried out with the setup shown in Fig. 4, with ADWISER inserted. The results are shown in Figs. 24 and 25

In Fig. 24, the 300–600 s interval shows the response times of Web-like downloads from the Internet server in the presence of a bulk *download* from the LAN server, without ADWISER. As

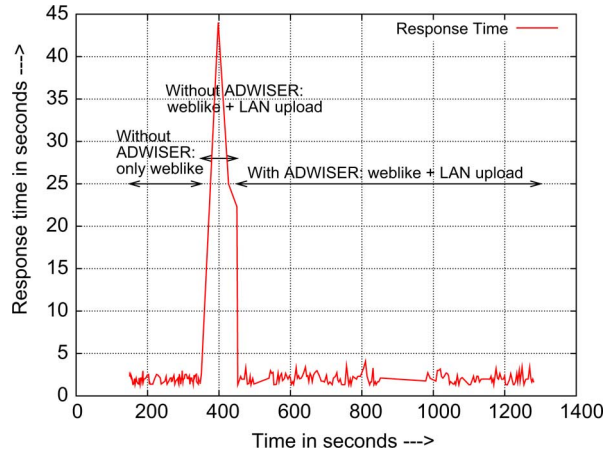

Fig. 25. Physical testbed: Response times of Web-like downloads in the presence of bulk upload to the LAN, with and without ADWISER.

TABLE I
SERVICE DIFFERENTIATION USING ADWISER

Configured Ratio	STA1 Thpt Mbps	STA2 Thpt Mbps	Total Thpt Mbps
5:1	3.879	0.776	4.655
9:1	4.190	0.465	4.655

can be seen, the average value of response time is 2.60 s for this interval. After introducing ADWISER at 600 s, we see that the response time improves, achieving a smaller average of about 2 s and also lower variability.

In Fig. 25, the 350–450 s interval shows the response time of Web-like download from Internet server in the presence of a bulk *upload* to the LAN server, without ADWISER. The average value of response time is 30.43 s. After introducing ADWISER at 450 s, we see that the response time improves with an average value of 2.05 s, which is a very significant improvement.

C. Policy-Based Service Differentiation

Consider two stations, STA1 and STA2, associated at 11 Mb/s, each downloading a large file. STA1 is a privileged user and requires some guaranteed fraction of the total throughput. ADWISER provides for such configurable policies. In Table I, we have presented two cases. In the first row, ADWISER is configured to give STA1 five times the throughput of STA2. In the second row, STA1 gets nine times the throughput of STA2. In ADWISER, a queue is created for each STA, and appropriate weights (recall the ϕ_i values from Section III-B) are assigned. We see from Table I that ADWISER is able to provide the required service differentiation. The total throughput in each case is the same: 4.655 Mb/s.

Fig. 26 shows the results of policy-based service differentiation on a testbed with a real AP. We have classified two STAs into the "gold" category, and two into the "bronze" category. All are associated at 54 Mb/s. All the four stations download bulk data from the Internet server. We use Linux *netem* to set an RTT of 300 ms for the gold stations and a slightly lower RTT of 200 ms for the bronze stations. In Fig. 26, in the interval 0–500 s, when ADWISER is bypassed, it is observed that the bronze STAs obtain a slightly higher throughput due

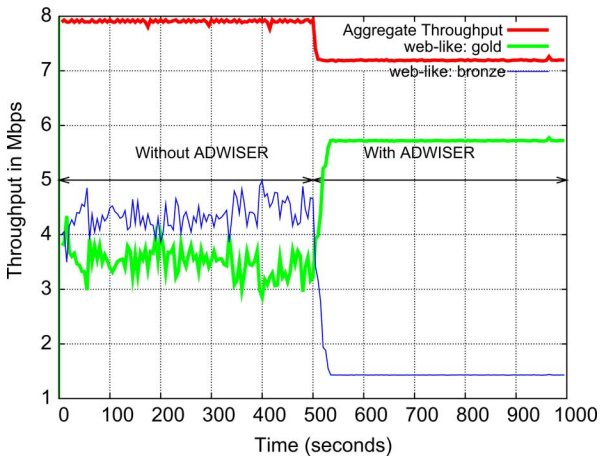

Fig. 26. Physical testbed: Enforcement of a throughput differentiation policy for Internet downloads using ADWISER. Aggregate throughput: uppermost thick line. Web-like-gold: thick line. Web-like-bronze: thin line.

to the lower RTT. However, from 500 to 1000 s, we have introduced ADWISER, with a policy 4:1. The virtual server in the WAN scheduler in ADWISER has a service rate that is 90% of the Internet access link speed (recall Section III-A); hence, the total throughput over the WAN is limited to 7.2 Mb/s when ADWISER is inserted. We observe that the total throughput of 7.2 Mb/s is shared in the ratio of 4:1 between the gold STAs and the bronze STAs.

Relation to the Literature: ADWISER supports policy-based service differentiation. This is a novel aspect that adds an important capability to traffic management in WLANs and that, to our knowledge, is not available in the open literature.

V. Conclusion

ADWISER, an enhanced version of WM [1], [2], is a centralized WLAN controller that can manage certain QoS issues in an infrastructure WLAN, without any modification to APs or the clients. These systems can be viewed as controllers in the "Split-MAC" architecture [10], [11]. We have reported our experiences in implementing ADWISER on a Linux platform and the performance results we have obtained. All the results were obtained from testbeds in which the actual physical ADWISER was in place and was carrying TCP traffic between a server and clients. We have reported that several TCP QoS problems in WLANs can be effectively solved by ADWISER. ADWISER can also be used to create any desired service differentiation between STAs.

Another experience we must report was that, in the course of evolving the ADWISER idea and experimenting with it, the insights gained from analytical models (such as [5] and [6], to list only a couple of many such papers in our bibliography) were invaluable in: 1) understanding, explaining, and predicting the experimental results; and 2) guiding the design of the algorithmic techniques.

In ongoing work, we are extending ADWISER to manage multiple APs, which is possible only because of the centralized nature of our solution, implemented outside the APs.

References

[1] M. Hegde, S. V. R. Anand, A. Kumar, and J. Kuri, "WLAN manager (WM): a device for performance management of a WLAN," *Int. J. Netw. Manage.*, vol. 17, pp. 155–170, Jan. 2007.

[2] M. Hegde, P. Kumar, K. R. Vasudev, S. V. R. Anand, A. Kumar, and J. Kuri, "Experiences with WM: A centralised scheduling approach for performance management of IEEE 802.11 wireless LANs," in *Proc. WISARD*, Bangalore, India, Jan. 2009, pp. 594–603.

[3] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, ANSI/IEEE Std 802.11, 1999.

[4] G. Berger-Sabbatel, F. Rousseau, M. Heusse, and A. Duda, "Performance anomaly of 802.11b," in *Proc. IEEE INFOCOM*, 2003, vol. 2, pp. 836–843.

[5] R. Bruno, M. Conti, and E. Gregori, "An accurate closed-form formula for the throughput of long-lived TCP connections in IEEE 802.11 WLANs," *Comput. Netw.*, vol. 52, pp. 199–212, 2008.

[6] G. Kuriakose, S. Harsha, A. Kumar, and V. Sharma, "Analytical models for capacity estimation of IEEE 802.11 WLANs using DCF for internet applications," *Wireless Netw.*, vol. 15, no. 2, pp. 259–277, Feb. 2009.

[7] Q. Wu, M. Gong, and C. Williamson, "TCP fairness issues in IEEE 802.11 wireless LANs," *Comput. Commun.*, vol. 31, no. 10, pp. 2150–2161, Jun. 2008.

[8] O. Bhardwaj, G. Sharma, M. Panda, and A. Kumar, "Modeling finite buffer effects on TCP traffic over an IEEE 802.11 infrastructure WLAN," *Comput. Netw.*, vol. 53, pp. 2855–2869, 2009.

[9] S. Harsha, A. Kumar, and V. Sharma, "An analytical model for the performance evaluation of multimedia applications EDCA in an IEEE 802.11e WLAN," *Wireless Netw.*, vol. 16, pp. 367–385, 2010.

[10] L. Yang, P. Zerfos, and E. Sadot, "Architecture taxonomy for control and provisioning of wireless access points (CAPWAP)," RFC 4118, Jun. 2005.

[11] P. Calhoun, M. Montemurro, and D. Stanley, "Control and provisioning of wireless access points (CAPWAP) protocol specification," RFC 5415, 2009.

[12] K. Cai, R. Lotun, M. Blackstock, J. Wang, M. Feeley, and C. Krasic, "A wired router can eliminate 802.11 unfairness, but it's hard," in *Proc. 9th ACM HotMobile*, 2008, pp. 49–54.

[13] G. Tan and J. Guttag, "Time-based fairness improves performance in multi-rate WLANs," in *Proc. USENIX Annu. Tech. Conf.*, Berkeley, CA, May 2004, pp. 269–282.

[14] A. J. K. Kashibuchi and N. Kato, "Channel occupancy time based TCP rate control for improving fairness in IEEE 802.11 DCF," *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, pp. 2974–2985, Jul. 2010.

[15] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, "Understanding TCP fairness over wireless LANs," in *Proc. 22nd Annu. IEEE INFOCOM*, Mar. 2003, vol. 2, pp. 863–872.

[16] A. Detti, E. Graziosi, V. Minichiello, S. Salsano, and V. Sangregorio, "TCP fairness issues in IEEE 802.11 based access networks," 2005 [Online]. Available: http://netgroup.uniroma2.it/Stefano_Salsano/papers/salsano-tcp-fair-wlan.pdf

[17] J. Ha and C.-H. Choi, "TCP fairness for uplink and downlink flows in WLANs," in *Proc. IEEE GLOBECOM*, 2006, pp. 1–5.

[18] F. Vacirca and F. Cuomo, "Experimental results on the support of TCP over 802.11b: an insight into fairness issues," in *Proc. Wireless On-Demand Netw. Syst. Services*, Jan. 2006, pp. 20–25.

[19] N. Blefari-Melazzi, A. Detti, I. Habib, A. Ordine, and S. Salsano, "TCP fairness issues in IEEE 802.11 networks: Problem analysis and solutions based on rate control," *IEEE Trans. Wireless Commun.*, vol. 6, no. 4, pp. 1346–1355, Apr. 2007.

[20] F. R. P. Starzetz, M. Heusse, and A. Duda, "Virtual flow queueing for improving TCP performance over IEEE 802.11 WLANs," in *Proc. WCNC*, 2008, pp. 2158–2163.

[21] D. Malone, D. J. Leith, A. Aggarwal, and I. Dangerfield, "Spurious TCP timeouts in 802.11 networks," in *Proc. 6th WiOpt*, Berlin, Germany, Apr. 2008, pp. 43–49.

[22] F. Keceli, I. Inan, and E. Ayanoglu, "TCP ACK congestion control and filtering for fairness provision in the uplink of IEEE 802.11 infrastructure basic service set," in *Proc. IEEE ICC*, 2007, pp. 4512–4517.

[23] S.-H. Yoo, J.-H. Choi, J.-H. Hwang, and C. Yoo, "Eliminating the performance anomaly of 802.11b," in *Proc. ICN*, 2005, LNCS 3421, pp. 1055–1062.

[24] H. Kim, S. Yun, I. Kang, and S. Bahk, "Resolving 802.11 performance anomalies through QoS differentiation," *IEEE Commun. Lett.*, vol. 9, no. 7, pp. 655–657, Jul. 2005.

[25] S. W. Kim, B.-S. Kim, and Y. Fang, "Downlink and uplink resource allocation in IEEE 802.11 wireless LANs," *IEEE Trans. Veh. Technol.*, vol. 54, no. 1, pp. 320–327, Jan. 2005.
[26] D. J. Leith and P. Clifford, "Using the 802.11e EDCF to achieve TCP upload fairness over WLAN links," in *Proc. 3rd IEEE WiOPT*, 2005, pp. 109–118.
[27] Scalable Network Technologies, Inc., Los Angeles, CA, "Scalable Networks," 2010 [Online]. Available: http://www.scalable-networks.com
[28] D. J. Leith, P. Clifford, D. Malone, and A. Ng, "TCP fairness in 802.11e WLANs," *IEEE Commun. Lett.*, vol. 9, no. 11, pp. 964–966, Nov. 2005.
[29] D. J. Leith and P. Clifford, "Modelling TCP dynamics in wireless networks," in *Proc. Int. Conf. Wireless Netw., Commun. Mobile Comput.*, Jun. 2005, vol. 2, pp. 906–911.
[30] Linux Wireless, "Rate control," 2012 [Online]. Available: http://linuxwireless.org/en/developers/Documentation/mac80211/RateControl/minstrel
[31] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 690–704, Oct. 1997.

Malati Hegde received the Ph.D. degree in mathematics from the Indian Institute of Technology, Kanpur, India, in 1978.

She is working as a Principal Scientific Officer with the Electrical Communication Engineering (ECE) Department, Indian Institute of Science (IISc), Bangalore, India. She is actively involved the Education and Research Network (ERNET) project, a nationwide academic and research network, and shares the responsibility of the ERNET NOC at IISc, Bangalore, in network planning and providing Internet services. Her areas of interest are network management and performance management in wired, wireless LANs, and wireless sensor networks.

Pavan Kumar received the B.Tech. degree in computer science from the National Institute of Technology Rourkela, Rourkela, India, in 2005.

He joined the Electrical Communication Engineering (ECE) Department, Indian Institute of Science (IISc), Bangalore, India, as a Project Assistant in 2007. Since 2010, he has been with Brocade Communications Systems, Bangalore, India, developing network protocol software for Ethernet switches.

K. R. Vasudev received the B.E. in electronics and communications from Visvesvaraya Technological University, Belgaum, India, in 2007, and the Master's degree in digital communications from the University of Kiel, Kiel, Germany, in 2012, and is currently pursuing the Ph.D. degree in speech and audio signal processing at the Digital Signal Processing and System Theory (DSS) chair at the Universtiy of Kiel.

He was with the Electrical Communication Engineering (ECE) Department, Indian Institute of Science (IISc), Bangalore, India, from 2008 to 2009 as a Project Assistant. His Master's thesis was on the topic of multichannel speech quality improvement for automotive systems.

N. N. Sowmya was born in Kerala, India, in 1986. She received the B.E. (Hons) EEE degree from the Birla Institute of Technology and Science, Pilani (Goa campus), India, in 2008.

She worked with Honeywell Technology Solutions, Bangalore, India, as a Software Engineer for two years. She was then a Project Assistant with the Electrical Communication Engineering (ECE) Department, Indian Institute of Science, Bangalore, India, for one year, during which she worked on the WLAN Manager/ADWISER project.

S. V. R. Anand received the B.E. degree in electrical communication engineering (ECE) from the Indian Institute of Science (IISc), Bangalore, India, in 1987.

Working as a Technical Consultant, he is involved in the implementation aspects of various sponsored R&D projects undertaken by the ECE Department, IISc. He is also associated with the ERNET Network Operations Centre, IISc. His areas of interests include network system software development, and operating systems.

Anurag Kumar (S'81–M'81–SM'92–F'06) received the B.Tech. degree from the Indian Institute of Technology, Kanpur, India, in 1977, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1981, both in electrical engineering.

He was then with Bell Laboratories, Holmdel, NJ, for over six years. Since, 1988 he has been with the Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore, India, where he is now a Professor and also Chair of the Electrical Sciences Division. From 1988 to 2003, he was the Coordinator of the Education and Research Network Project (ERNET), IISc, India's first wide-area packet switching network. He is a coauthor of the graduate textbooks *Communication Networking: An Analytical Approach* (Morgan Kaufmann, 2004) and *Wireless Networking* (Morgan Kaufmann, 2008) by A. Kumar, D. Majunath, and J. Kuri.His area of research is communication networking; specifically modeling, analysis, control, and optimization problems arising in communication networks and distributed systems.

Prof. Kumar has been elected Fellow of the Indian National Science Academy (INSA), the Indian National Academy of Engineering (INAE), and the Indian Academy of Sciences (IASc). He was an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING and an Area Editor of *IEEE Communications Surveys and Tutorials*.

Joy Kuri received the B.E. degree in electronics and telecommunication engineering from Jadavpur University, Kolkata, India, in 1987, and the Ph.D. degree in electrical communication engineering (ECE) from the Indian Institute of Science (IISc), Bangalore, India, in 1995.

Subsequently, he spent two years with Ecole Polytechnique, University of Montreal, Montreal, QC, Canada, and one and a half years with INRS-Telecommunications, University of Quebec, Montreal, QC, Canada, as a Research Associate. Since 1999, he has been with the Centre for Electronics Design and Technology, IISc, where he is now a Professor. His research and teaching interests are in the areas of modeling, analysis, and control of communication networks and stochastic systems.