NAPman: Network-Assisted Power Management for WiFi Devices

Eric Rozner*
Univ. of Texas at Austin
Austin, TX 78712, USA
erozner@cs.utexas.edu

Vishnu Navda
Microsoft Research India
Bangalore 560080, India
navda@microsoft.com

Ramachandran Ramjee
Microsoft Research India
Bangalore 560080, India
ramjee@microsoft.com

Shravan Rayanchu*
Univ. of Wisconsin-Madison
Madison, WI 53706, USA
shravan@cs.wisc.edu

ABSTRACT

WiFi radios in smart-phones consume a significant amount of power when active. The 802.11 standard allows these devices to save power through an energy-conserving Power Save Mode (PSM). However, depending on the PSM implementation strategies used by the clients/Access Points (APs), we find *competing background traffic* results in one or more of the following negative consequences: a significant increase, up to 300%, in a client's energy consumption, a decrease in wireless network capacity due to unnecessary retransmissions, and unfairness.

In this paper, we propose NAPman: Network-Assisted Power Management for WiFi devices that addresses the above issues. NAPman leverages AP virtualization and a new energy-aware fair scheduling algorithm to minimize client energy consumption and unnecessary retransmissions, while ensuring fairness among competing traffic. NAPman is incrementally deployable via software updates to the AP and does not require any changes to the 802.11 protocol or the mobile clients. Our prototype implementation improves the energy savings on a smart-phone by up to 70% under varied settings of background traffic, while ensuring fairness.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.4 [**Performance of Systems**]: [Design studies]

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

*The author was an intern at Microsoft Research India during the course of this work.

MobiSys'10, June 15–18,2010, San Francisco, California, USA

Keywords

Wireless Local Area Networks (WLANs), Access Points (APs), Smart-Phones, Power Save Mode (PSM), Scheduling, 802.11

1. INTRODUCTION

Wireless communication imposes a significant energy cost on mobile smart-phones. For example, the base energy consumption of the HTC Tilt 8900 series phone ranges between 155-475mW, depending on the intensity of the backlight. In comparison, the WiFi radio consumes over 1000mW while transmitting. Thus, optimizing WiFi power consumption is essential for maximizing the battery life of mobile smart-phones.

WiFi radios support power saving mechanisms by implementing multiple modes of operation with different power and performance characteristics. For example, the Tilt's WiFi radio's *active* or Constantly Awake Mode (CAM) draws high power (1120mW) and delivers low latency, while the radio in *sleep* or Power Save Mode (PSM) consumes little power (72mW) at the cost of increased latency.

IEEE 802.11 static and adaptive PSM mechanisms and related research literature [2, 16, 17, 19, 27] include a number of techniques for leveraging PSM in order to save energy (Section 2). In these approaches, the 802.11 Access Point (AP) supports PSM by 1) buffering incoming packets for WiFi clients in PSM, 2) indicating the presence of buffered packets via fields in beacon messages, and 3) delivering the buffered packets after the client notifies the AP it's ready to receive one or more packets. This allows the WiFi radio to spend most of the time in low power PSM, waking up to high power CAM only in order to receive its intended packets.

However, depending on the PSM implementation strategies used by the APs and the clients, we find through controlled experiments that *competing background traffic* results in one or more of the following negative consequences: a significant increase in the client's energy consumption, a decrease in wireless network capacity due to unnecessary retransmissions, and unfairness (Section 4). Furthermore, our network trace analysis indicates that such situations are likely quite common during peak usage.

In this paper, we present NAPman, a Network-Assisted Power Management solution that minimizes WiFi energy consumption for mobile devices, even in the presence of competing traffic. NAPman implements a new energy-aware fair scheduling algorithm at the AP that minimizes WiFi radio wakeup time and eliminates unnecessary retransmissions in the presence of competing traffic. Further, NAPman leverages AP virtualization in a novel manner in order to

isolate PSM clients from each other. Finally, NAPman is incrementally deployable since it requires only software updates to the AP — no changes are needed to IEEE 802.11 Standards or the WiFi devices/smart-phones.

In order to minimize energy consumption in WiFi devices, NAPman solves two key challenges: isolation of PSM clients from competing a) CAM traffic and b) other PSM traffic. Let us first consider the problem of competing CAM traffic. Through controlled experiments, we found different commercial APs implement different scheduling strategies when a PSM client wakes up and notifies its AP. Many implementations simply perform *normal scheduling* that enqueues all buffered packets of a PSM client to the tail of the transmit queue. This increases the time PSM clients remain in high power CAM, wasting energy, while packets that were ahead in the queue are being transmitted. To deal with this issue, different phones implement different PSM strategies. For example, phones such as the iPhone 3GS use adaptive PSM with aggressive timeout values of $20-25$ms. When these clients encounter APs that use normal scheduling, the net effect is the client sleeps quickly while its PSM packets in the transmit queue end up being retransmitted multiple times. When these "losses" are coupled with rate adaptation, they result in significant waste of network capacity (Section 4.2). Other commercial APs implement a *high priority* solution for PSM clients, whereby they simply enqueue the buffered PSM packets to a higher priority queue. While this approach helps save energy when one PSM client is competing with CAM clients, we show such a simplistic approach can result in significant unfairness to other CAM clients (Section 4.3).

The NAPman energy-aware fair scheduler aims to deliver energy savings without unfairness. A simple fairness policy is to deliver packets based on the order of arrival, in a first-come-first-served (FCFS) manner. However, FCFS policy turns out to be non-work conserving, as packets destined to PSM clients cannot be delivered until the client wakes up. The NAPman scheduler enforces a work-conserving FCFS policy where the FCFS constraint is applied only to packets of clients that are awake at any given time. In NAPman, the AP advertises the presence of buffered packets for the PSM client in the upcoming beacon only if high priority scheduling of at least one buffered packet of the PSM client does not 'skip-ahead" of any packet that has been waiting longer in the transmit queue. Unlike AP implementations today, the scheduler continues to maintain a queue of buffered packets for adaptive PSM clients and manages their packet delivery carefully. These mechanisms allow both static and adaptive PSM clients to quickly go back to sleep, saving energy. Interestingly, we find the NAPman scheduler not only saves energy, but also reduces latency compared to normal scheduling since adding PSM packets to the tail of the queue tends to be unfair to PSM clients.

While the fair scheduling approach helps isolate PSM from CAM traffic, the presence of multiple PSM clients associated with a single AP is still problematic since the PSM clients may need to stay awake while other PSM clients are being served. In this case, both normal and high priority scheduling result in increased client energy consumption. Instead, NAPman relies on a novel solution that leverages virtualization. The NAPman AP advertises several virtual APs through beacons that are staggered in time. The PSM clients are then made to associate to the appropriate virtual AP, thereby isolating PSM clients from each other.

We have implemented the NAPman scheduling and virtualization mechanisms in the MadWifi driver (Section 6). Through extensive experiments under a variety of traffic conditions (Section 8), we show that NAPman delivers energy savings similar to or better than the high priority solution, avoids unnecessary retransmissions, and enforces fairness for both PSM and CAM clients.

Finally, for completeness, we design an extension to NAPman that is able to take advantage of simple changes on the clients in order to provide further energy and latency benefits (Section 9).

In summary, the contributions of the paper are as follows:

- Through controlled experiments and wireless network trace analysis, we show that current AP and client PSM implementations can negatively impact energy consumption, network capacity, and/or fairness in the presence of competing traffic.

- We present the design and implementation of NAPman, an AP-based incrementally deployable solution that utilizes a new energy-aware fair scheduling algorithm and leverages virtualization to minimize energy consumption and unnecessary retransmissions for static/adaptive PSM clients while maintaining fairness.

- Using extensive experiments, we show that NAPman is able to deliver significant energy savings of up to 70% compared to current AP implementations for various network traffic workloads.

2. BACKGROUND AND RELATED WORK

We start with a brief background of the power saving mechanisms available in the WiFi standard and then discuss related research proposals.

2.1 Background

WiFi radios typically support multiple modes of operation, with varying power consumption in each mode. As mentioned earlier, most WiFi devices utilize at least two power modes: a high-power Constantly Awake Mode (CAM) with best performance and an energy-conserving Power Save Mode (PSM), where the radio periodically wakes up to receive data.

In order to take advantage of low power PSM, a *static PSM* approach was originally standardized [16]. The AP buffers packets for clients in PSM and indicates, through the Traffic Indication Map (TIM) fields in the beacon, the presence of buffered packets for clients. The client device wakes up for beacons and if the device's corresponding TIM field is set, it sends a separate PS-POLL message to receive each buffered packet. A MORE bit in the data frame indicates if more packets are buffered at the AP, helping the client decide when to stop sending PS-POLL messages.

While the static PSM approach allows the device to save power by only waking up when packets are outstanding at the AP, the latency involved in receiving packets via PS-POLL has been found to be high for interactive applications such as web browsing [19]. Therefore, many WiFi devices today also implement a technique known as *adaptive PSM* [19], where the device switches between PSM and CAM based on some heuristics (e.g. TIM bit set, reception of a threshold number of packets or lack of network activity for a pre-defined duration). The device notifies the AP of its transitioning to PSM or CAM by sending NULL data frames with the power management bit set to 1 or 0, respectively.

The adaptive PSM approach allows the device to be in CAM while the user is browsing, and thus interactive performance of adaptive PSM is as good as CAM. However, since the device remains in CAM for an idle timeout period after every PSM to CAM transition, for many background applications with intermittent network activity, PSM adaptive can be more expensive in terms of energy consumption than PSM static.

The NAPman approach is complementary to both the static and adaptive PSM approaches since it minimizes the time the device needs to stay in high power CAM by isolating it from competing traffic. Depending on the values used as part of heuristic-based triggers, the energy savings of NAPman will vary for adaptive PSM.

Finally, in order to enable power savings while supporting QoS sensitive applications such as VoIP, the Unscheduled Automatic Power Save Delivery (U-APSD) is defined by the IEEE 802.11e standard [17]. In this approach, whenever a device sends a frame to the AP (actual data or a NULL trigger frame), the AP immediately sends back buffered frames scheduled as high priority and in a burst, using the 802.11e TXOP mechanism. In a typical VoIP application where a device can anticipate receiving periodic frames (e.g., every 20ms), the U-APSD approach results in no added latency while still allowing the device to sleep between frames.

2.2 Related Work

There has been a significant amount of research effort devoted to power savings for WiFi radios. We classify these efforts into four broad categorizes: sleep optimization, impact of background traffic on PSM, network support for energy saving and use of side-channel information, and describe them below while placing our contributions in context.

2.2.1 Sleep Optimization

A lot of work has focused on improving the heuristics in adaptive PSM [2, 19, 27]. The authors in [19] and [27] design adaptive PSM techniques, motivated for applications such as web browsing, that provide bounded delay while minimizing energy. In [2], application characteristics are identified using hints, e.g. background versus foreground traffic, and these hints are used to automatically switch between WiFi PSM and CAM in order to minimize energy consumption without sacrificing performance. Another approach [6], specifically targeted towards streaming media, powers down or powers up the WiFi interface when the application playout buffer is full or almost empty, respectively, thereby saving energy without impacting application performance.

In [26], the authors present an adaptive U-APSD approach that extends the 802.11e-based U-APSD to applications such as web browsing or file download. The device tries to predict when buffered packets may be available at the AP and sends NULL trigger frames to retrieve the packets. This approach can result in the device transmitting unnecessary NULL trigger frames and experiencing associated energy wastage. Furthermore, the use of bursty high priority transmissions for applications with large amounts of data such as browsing or file download can lead to unfairness or starvation, an issue not considered in [26].

In [7], the authors propose a technique called forced idling that puts the radio in a low power idling state, as an alternative to sleep mode, to avoid wasting energy due to overhearing background communications. The approach relies on overhearing RTS/CTS frames for identifying the time duration to idle. However, since most WiFi deployments turn off RTS/CTS due to its high overhead [32], the practicality of this approach is limited.

In [23], the authors propose micro power management (μPM), a client-based solution that allows a WiFi radio to sleep for very short intervals, such as a few microseconds, which can be used to sleep even between two MAC frames. μPM uses prediction to exploit short idle intervals and does not need any special support from the AP, since it relies on 802.11 retransmissions to recover from any mispredictions. NAPman is focused on fair and energy-efficient scheduling at the AP that isolates client's traffic from each other and is thus complementary to μPM.

2.2.2 Impact of Background Traffic

In Scheduled PSM [13, 15], it is observed that background PSM unicast and multicast traffic can result in energy drain on static PSM clients. The authors refer to competing PSM flows as background traffic relative to a selected PSM client [14], whereas we generalize the notion of background traffic to include other CAM clients associated to an AP. Scheduled PSM overlays a TDMA-like structure over 802.11 whereby the beacon period is divided into *time slices* and each PSM client's packets are delivered only in its advertised time slice. This is achieved by changing the TIM field in the 802.11 standard to include new slicing control and slicing map fields that indicate the number and offset of the time slices in the beacon interval. At the beginning of each time slice, the AP takes control of the channel by using either RTS/CTS or Self-CTS, and schedules traffic for the appropriate PSM client during that interval. The authors demonstrate that their modifications result in energy savings through simulation using varying rates of UDP traffic to PSM clients. However, this solution requires modifications to the 802.11 standard and, thus, changes to both mobile clients and APs are necessary. Further, this solution does not consider fairness issues and does not support adaptive PSM clients.

In this paper, we show through extensive experiments that background unicast traffic can negatively impact not only the energy drain of PSM clients, but also fairness to CAM clients and even network capacity. Further, our AP-based solution, NAPman, does not require any changes to the 802.11 protocol and can reduce the energy consumption of unmodified static and adaptive PSM clients in the presence of background CAM or PSM traffic.

2.2.3 Network Support

Network-based proxies are also often used for reducing application energy consumption [4, 9]. In [4], the polling responsibility of applications is shifted from the mobile device to a network-based proxy which then aggregates and sends the poll responses in a batch. In [9], a proxy is used to batch packets from various streaming applications in a coordinated manner with the mobile device so that the device can sleep between the receptions of batched packets. In PSM-throttling [31], the authors utilize traffic shaping for energy reduction without using a proxy by increasing the burstiness of traffic from a streaming server using targeted zero-sized TCP receive window messages.

While these approaches help shape a given client's traffic to save energy, NAPman isolates client's traffic from each other to save energy, and therefore these techniques are complementary to NAPman. Other work prioritizes a subset of PSM clients through service level contracts between clients and access points [20], but does not try to eliminate PSM client contention.

Lastly, a set of approaches allows the AP to help its clients save power. In [30], the authors propose multiple-bit TIMs that indicate the number of packets for each PSM node to receive. Assumptions are made on PSM traffic (sent in contiguous bins, fixed packet size, etc.), allowing nodes to sleep during the transmission to other PSM nodes. In Centralized PSM [33], the AP selects certain parameters for its clients, such as beacon interval, listen interval and contention window size, in order to reduce the simultaneous wake-ups of clients. In LAWS [22], the AP advertises a subset of PSM clients in the beacon and clients use information in the beacons to determine their polling sequence in order to help avoid client contention. Finally, [21] uses heuristics to approximate the global optima in power-savings over all PSM clients in a generalized PSM setting. All of these approaches require both client and AP modifications. In NAPman, only AP-side changes are required to isolate

Access Point	PSM Scheduling
Linksys BEFW11S4 ver 4	Normal
MadWifi 0.9.4-based AP	Normal
D-Link DIR-635 802.11n	High priority
DD-WRT v23 SP2-based AP	High priority
Linksys WRT54GL	Normal
Linksys WRT310N 802.11n	High priority
Netgear WGR614 v7	Normal

Table 1: PSM scheduling techniques for APs in the wild

PSM clients from one another and to isolate PSM traffic from background traffic in a fair manner.

2.2.4 Side-Channel Information

Given the high energy costs of WiFi scanning and CAM operation, several systems have been proposed to turn on the WiFi radio based on some side-channel information. Wake-on-Wireless [29] uses a low power radio to turn on WiFi, while Cell-to-Notify [1] uses incoming cellular caller ID information to wake up the WiFi radio. Further, context information such as current location from cellular towers or usage history [28] or even Bluetooth contact patterns [3] can be used to avoid unnecessary WiFi scanning. Again, these approaches are complementary to NAPman.

3. PSM IMPLEMENTATION AT AP

We would like to first understand how PSM is implemented in existing commodity APs. Since this is usually not described in the vendor provided specifications, we ran our own controlled experiments where we connect some PSM and CAM clients to the AP, and examine how the AP delivers PSM packets while varying background traffic to the CAM clients. We use WiFi equipped smartphones as clients for this experiment (details of the experimental setup can be found in Section 7) and embed sequence numbers in the payload of each packet. We then analyze the wireless packet traces and try to reverse engineer the scheduling mechanism implemented at the AP.

3.1 PSM Implementation on Commercial APs

We found that most APs implemented PSM packet delivery in one of two ways: *normal* or *high priority*. In the case of normal scheduling, on receiving a PS-POLL packet or a NULL frame from a PSM client, the AP enqueues one or more PSM packets at the tail of the transmission queue. In the case of high priority scheduling, the buffered PSM packets are transmitted immediately by queueing the packets in a separate transmission queue with a higher priority. Table 1 lists some of the popular commodity APs we tested and the PSM implementation mode used.

The PSM implementation can impact client energy consumption (Section 4.1), network capacity (Section 4.2) and fairness (Section 4.3). We next discuss how normal and high priority implementations are unfair to PSM and CAM clients, respectively. We then propose a fairness property that is desirable and is enforced by the NAPman PSM implementation.

3.2 Fairness

The normal implementation queues PSM traffic at the transmission queue's tail and is unfair to PSM clients because their packets may be transmitted by the AP after packets to other CAM clients, even if the packets to CAM clients arrived subsequent to the PSM packets (the PSM packets may have been waiting in the PSM buffer).

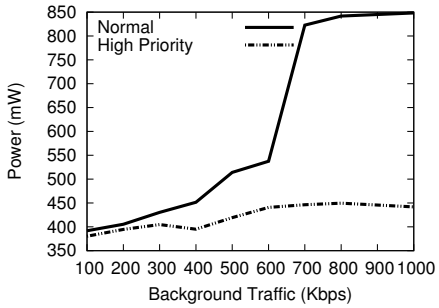

Figure 1: Power drawn by a static PSM client receiving a 128 Kbps radio stream with varying CBR background traffic

On the other hand, the high priority implementation can be similarly unfair to CAM clients because PSM packets are scheduled at the head of the queue. Thus, current commercial AP implementations are unfair to either PSM or CAM. This raises the question: What is a good fairness policy to enforce at the AP?

Clearly, while first-come-first-served (FCFS) is a good fairness policy, scheduling packets in a FCFS manner can be non-work conserving. A strict FCFS policy would require CAM packets that arrive when an older PSM packet is buffered to unnecessarily wait until the PSM packet is sent. This idle time can waste network capacity, and thus a strict form of FCFS is not desirable.

In this paper, we enforce a modified form of FCFS that is work conserving. Basically, we ensure that packets are FCFS for all clients (CAM/PSM) that are awake. When a PSM client is asleep, packets for other CAM (and awake PSM) clients can *go ahead* of the packets of the sleeping PSM client, thereby avoiding wastage of network capacity. However, when a PSM client is awake, its packets will be scheduled FCFS – ahead of packets from other CAM (or awake PSM) clients that arrived later, but behind packets from other CAM (or awake PSM) clients that arrived earlier.

Ensuring FCFS alone may not be sufficient at times, since it can result in temporary starvation for clients in an adversarial setting. Consider a PSM client which wakes up infrequently to download buffered PSM packets. Whenever the PSM client wakes up, there could be multiple buffered packets that are fair in the FCFS sense, and all such packets are scheduled for transmission in a single burst. Note that this is a problem for the normal and high priority scheduling approaches as well. This problem can be mitigated by a combination of aging packets in the queue and putting a limit on the number of packets that are buffered in the queue.

4. MOTIVATION

In this section, we motivate the problem with today's PSM implementations in the presence of competing traffic.

We first highlight, through controlled experiments, the three serious drawbacks that exist in current PSM implementations in the presence of competing background traffic, namely, significant increase in client's energy consumption, decrease in wireless network capacity due to unnecessary retransmissions and unfairness. Finally, we use real network traces obtained from a large scale wireless network [18] as the background traffic and evaluate its impact on a hypothetical PSM client attached to that network.

4.1 Impact on Energy Consumption

Single Static PSM Client: Consider the case of a single PSM client attached to the AP that is subjected to varying amounts of background traffic. The normal delivery scheme at the AP ensures

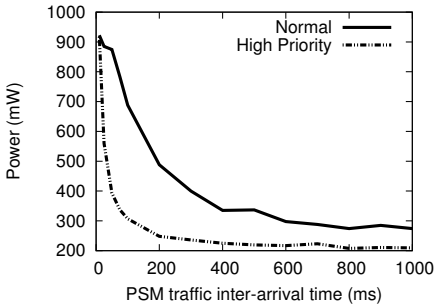

Figure 2: Power drawn by a static PSM client with saturated background traffic and varying PSM packet inter-arrival time

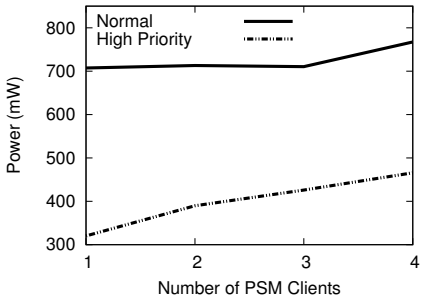

Figure 3: Power drawn by a static PSM client (1 pkt/100ms) with varying PSM clients and saturated CBR background rates

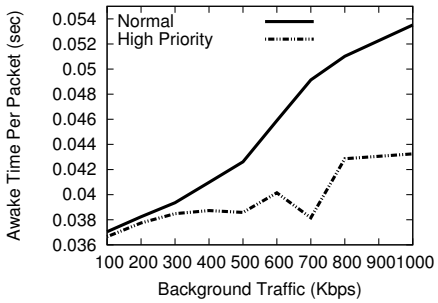

Figure 4: Per-packet awake time for an iPhone 3GS receiving a 128 Kbps radio stream with varying CBR background rates

the transmission of a buffered PSM packet will never precede packets that arrive earlier than the buffered packet at the AP. While this ensures that PSM packets are not unfair to other packets, this has negative implications in terms of energy savings when there is a significant number of packets destined for other clients in the transmission queue. In this case, the PSM client is forced to remain in high power CAM mode for a long duration, while background packets that are already in the queue are being drained, before receiving its buffered PSM packet. Note that, for this case, the high priority scheme doesn't suffer from this problem since the PSM client will receive its buffered packets immediately as they are given priority over other packets already in the transmission queue.

Figure 1 depicts the base power consumed by an 802.11b static PSM client (an HP iPAQ hw6945) that is receiving a 128 Kbps radio stream as we vary the rate of a background CBR UDP flow to a CAM client associated to the same AP. We fix the AP's bitrate to 1 Mbps in this experiment for simplicity (see Section 7 for our methodology). The power consumption starts to increase sharply for the normal scheduling scheme at 400 Kbps CBR rate (almost 50% channel utilization) as the transmission queue starts to build up with background packets. When the *channel is saturated with background traffic, the power consumption increases to nearly 2.5 times the no background traffic case*. On the other hand, as expected, the background traffic has negligible impact on power consumption when the AP is using the high priority scheme.

The impact on PSM client power consumption depends on the inter-packet arrival time of the PSM packets. Figure 2 depicts the power consumed by a PSM client when the background traffic is saturated as we vary the inter-packet arrival time for PSM packets from one 1024 byte packet every 10ms to one every second. In the case of one packet every 10ms, both schemes essentially stay awake continuously because the PSM traffic is saturated since the data rate is fixed at 1Mbps. We see that for inter-arrival times between 50-200ms, high priority results in up to 55% lower power consumption and even for the case of only one PSM packet per second, the high priority scheduling results in 25% lower power consumption than normal scheduling. However, as we shall see next, high priority scheduling is not effective when there are multiple PSM clients.

Multiple Static PSM Clients: Consider the case of multiple PSM clients attached to the same AP. Figure 3 plots the power drawn by one PSM client as the number of PSM clients are increased and background traffic is at saturation. The load for PSM clients is one 1024 byte packet every 100ms. In this case, not only does normal PSM scheduling cause power consumption to increase by 2X, but *even the high priority scheduling results in increased power consumption of up to 45% for the PSM clients*. This is because the high priority scheme is unable to isolate the PSM clients from each other. Thus, PSM clients end up remaining in high power CAM while packets of other PSM clients are drained from the high priority queue. Note that normal scheduling does not see the same increase in this case because under high background loads the mobile nodes are awake for most of the time anyways.

Adaptive PSM Client: We now examine how background traffic impacts clients that implement adaptive PSM using NULL frames. Recall that adaptive PSM clients go from CAM to PSM when they do not receive any packets for an idle timeout interval. Different client implementations choose different values for the timeout interval. If the timeout interval is larger than the packet inter-arrival time, the client remains in CAM throughout the duration of the communication, irrespective of whether there is background traffic or not. For example, we found the HTC Magic phone running Android uses adaptive PSM with an idle timeout of approximately 3 seconds while the iPhone 3GS uses adaptive PSM with a timeout value in the range of 20-25ms. Thus, for an Internet radio application, the WiFi radio on HTC Magic would be in CAM continuously, while the WiFi radio on the iPhone would be able to sleep in between packet arrivals.

In Figure 4, we depict the impact of background traffic on the WiFi radio awake time of an iPhone 3GS client using the default settings on a 128 Kbps radio stream (see Section 7 for why we were unable to accurately measure the power consumption). In this case, we see that background traffic results in an increase of less than 50% in the radio awake time even under saturated conditions. Upon closer examination, we find that this somewhat limited impact of background traffic is because the iPhone client goes to sleep aggressively, which has a negative impact on the capacity of the wireless network as we discuss next.

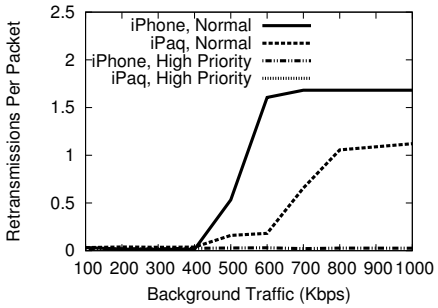

Figure 5: Average retransmissions per packet for a 128 Kbps radio stream to an iPhone and an iPAQ with varying CBR background rates

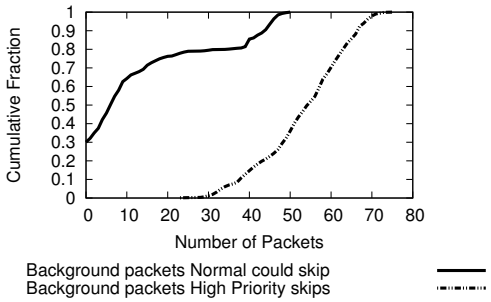

Figure 6: Examining fairness for a static PSM client (1pkt/100ms) with saturated background rate for different scheduling schemes

4.2 Impact on Wireless Network Capacity

One of the drawbacks of having an aggressive timeout value for transitioning the WiFi radio back to PSM is the risk of missing packets that are waiting for transmission in the transmit queue. These packets are invariably retransmitted multiple times by the AP when normal scheduling is employed, as can be seen from Figure 5. While the iPhone client manages to receive most of these packets eventually through retransmissions, these retransmissions are a significant and unnecessary burden, reducing the capacity of the wireless network. Even the iPAQ, which uses static PSM, suffers from unnecessary retransmissions, but its retransmissions are lower compared to the iPhone. Upon analysis of the wireless trace, we found that when the client does not receive a packet after sending a PS-POLL for the entire duration of a beacon, it examines the TIM bit in the next beacon and if it is not set, the client simply goes to sleep assuming the packet may have been lost. However, the packet is merely delayed due to high background load, and ends up being retransmitted many times.

We found one interesting observation on iPhones when we increased the inter-arrival time between PSM packets under saturated background load. We noticed that the number of retransmissions started to increase, and eventually reached the maximum retry limit per packet at one second inter-arrival time. This happens because a buffered packet is enqueued at the end of the transmission queue and the iPhone only waits 25ms (its idle timeout value) for the PSM packet. The PSM packet is not delivered in this time and the iPhone goes to sleep. The TIM bit in the subsequent beacons is not set (because there is no new PSM traffic) and the iPhone continues to sleep. As a result the buffered packet is eventually sent while the iPhone is sleeping and the packet incurs the maximum retransmissions before being dropped.

Another unfortunate side-effect of these retransmissions is the coupling with the auto-rate mechanism in the AP that ends up lowering the data rate of traffic to the client (and to the network as well due to the rate anomaly problem of 802.11 [5]), as we evaluate in Section 8. Increasing the timeout value would reduce retransmissions, but at the cost of increased energy consumption on the client.

While high priority scheduling mostly solves the retransmission issue, it has a major impact on fairness as discussed next.

4.3 Impact on Fairness

The *high priority scheme can be significantly unfair to the background traffic* as PSM packets skip ahead of packets that arrived earlier at the AP. The impact of unfairness on background flows becomes prominent when the PSM traffic in the network increases and can lead to starvation of background flows. On the other hand,

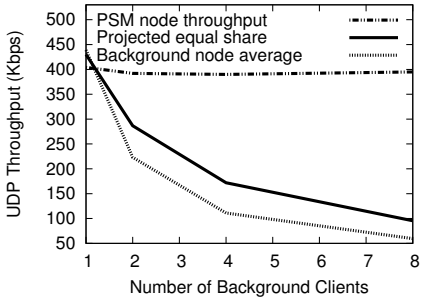

Figure 7: A single static PSM client with high-priority scheduling can take an unfair share of the WLAN capacity

the normal scheme is unfair to PSM clients since it enqueues PSM packets at the tail of the transmit queue even though these PSM packets might have arrived earlier than packets ahead of it in the queue.

Consider the case of a PSM client receiving 1 packet every 100ms subjected to saturated background CBR flow. Figure 6 shows a CDF of how many background packets are unfairly skipped by a PSM packet in the case of high priority implementation and how many newer background packets are scheduled ahead of a PSM packet in the case of normal scheduling. In the median case, about 54 background packets are unfairly skipped in the high priority scheme, while 6 newer background packets are ahead of the PSM packet in the normal scheme, demonstrating the unfairness inherent to both the scheduling schemes.

In Figure 7, we show how a single PSM client can adversely impact the available capacity of the network. We have one static PSM client and we vary the number of background clients. All clients, including the PSM client, receive saturated CBR traffic in the downlink direction. We plot the throughput of the single PSM client, the average of each background client, and a projected "equal" throughput, corresponding to what every node would have received if it received the same share of the medium. We see a single PSM client with high demand takes away half of the network capacity, leaving all the background nodes to share the remaining half. The reason the PSM client doesn't get full capacity is because the PS-POLL message must contend with the AP. If the PS-POLL gets the medium, then a PSM packet is sent. If the AP wins contention, then a background packet is sent. In the case of adaptive PSM (not shown), where the client sends only a single NULL frame to receive all its buffered packets, the client could use up almost the

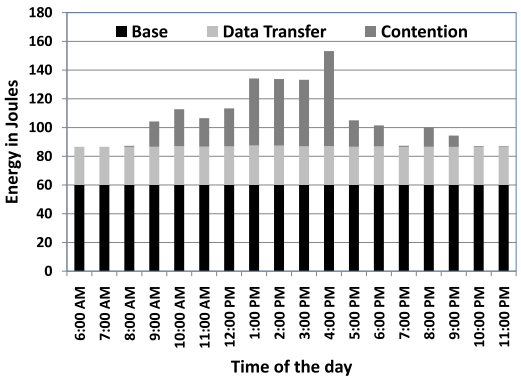

Figure 8: Impact of background network traffic on energy consumption of a PSM client streaming 192 Kbps radio

entire capacity of the network, starving all other clients, if the high priority scheme is used for adaptive PSM.

4.4 Trace Analysis

While the previous section highlighted the issues with current PSM implementations when subjected to controlled amounts of background traffic, it is not immediately clear if such situations are common in regular usage. In order to evaluate the impact of realistic network traffic on the energy savings of a PSM client, we conduct a trace based analysis using real network traces that were captured in the CSE department building at UCSD [18] for one full day in 2007. We use these traces as the background network traffic for a hypothetical PSM client attached to the same network. The published traces consist of both wired traces captured at a gateway router and wireless traces built by merging multiple trace files captured from sniffers placed at different locations in the building. In addition, the packet timestamps in these traces are precisely synchronized to a global clock, which is required for our analysis described below.

We built a simple simulator that models the PSM buffers and the transmission queue for each individual AP in the network. We then replay the UCSD traces in the simulator to recreate the state of the transmission queue in time with fine-grained detail. The timestamp of a packet in the wired trace is noted as the arrival time at the AP and the timestamp for the same packet in the wireless trace is taken as the departure time. In addition to replaying the background traffic, we simulate a PSM client that is connected to one of the APs in the network, and used a snippet of a real application trace to simulate the PSM traffic for this client.

We implement the PS-POLL based packet delivery scheme in the simulator and the AP follows the normal mode of operation where the buffered PSM packet is enqueued at the tail of the AP's transmission queue whenever a PS-POLL packet is sent by the client (given that the high priority mode has unfairness/starvation issues, we do not consider it a practical solution). We keep track of sleep and wakeup times for the PSM client while replaying the application trace. The total energy consumed by the PSM client for the duration of the application trace is calculated by multiplying the sleep and the wake up durations with the corresponding power consumption values obtained from our measurements. In order to analyze the impact at different background loads (and time of day), we vary the starting time for the PSM traffic trace relative to the background traffic trace.

Figure 8 shows the energy consumed by the PSM client at the end of a 10 minute trace of an 192 Kbps Internet radio application streaming, when replayed at different starting times in the 24 hours of background network trace. During the off-peak hours when the network is lightly loaded, the client does not incur any delay in receiving the buffered PSM packets and thus sees no negative impact on energy usage. However, *during peak hours we see that the energy consumption of the PSM client is up to 75% higher*, due to contention with background traffic. This analysis validates our premise that competing background traffic, as seen in realistic traces, can indeed cause significant energy drain on PSM clients.

Client-side PSM	AP-side PSM Scheduling	
	Normal	High Priority
Static	Low energy savings	Unfairness
Adaptive (aggressive timeouts)	Reduced capacity	Unfairness
Adaptive (long timeouts)	Low energy savings	Low energy savings Unfairness

Table 2: Summary of problems for different AP and client implementations of PSM

4.5 Summary

Using controlled experiments, we first reverse engineered the PSM scheduling implementation in several commercial APs and identified APs implement either normal scheduling where PSM packets are enqueued to the tail of the transmission queue or high priority scheduling where PSM packets are sent as high priority. We then showed when there is one PSM client attached to the AP, competing background traffic can cause PSM client power consumption to increase up to 3X with normal scheduling. While high priority scheduling keeps power consumption unchanged for the case of one PSM client amidst background traffic, power consumption of PSM clients goes up by 45% with high priority scheduling when four PSM clients are attached to the same AP. We then illustrated the negative impact on network capacity due to unnecessary retransmissions that are caused because of WiFi clients aggressively sleep. In terms of fairness, we found normal scheduling is unfair to PSM clients, while high priority scheduling can result in significant unfairness to background traffic. Table 2 summarizes all the problems we identified for different combinations of AP and client implementations of the PSM protocols. Finally, using real network traces [18] as background traffic, we show that energy consumption of the PSM client can increase by as much as 75% during peak hours.

We now present the design of NAPman that addresses the issues identified above, i.e., NAPman will provide fair scheduling at the AP while, at the same time, ensuring that energy consumption of one or more PSM clients is not negatively impacted by background traffic.

5. NAPMAN SYSTEM

In this section, we first enumerate the design constraints for implementing an effective PSM solution on the AP. Next, we go on to describe NAPman, an AP-based system that addresses the drawbacks of existing solutions, while meeting our design constraints.

5.1 Design Constraints

We now focus on the design constraints of a system for efficient delivery of PSM packets buffered at the AP, which can assist PSM clients to effectively conserve energy even in the presence of competing background traffic. To be effective, yet practical, the system needs to meet the following constraints. First, the energy savings should not be sacrificed at the expense of eliminating fairness,

since unfairness can adversely impact the performance of non-PSM clients. Second, changes to the clients should not be required since such a solution would be difficult to deploy given there are plethora of very diverse WiFi clients already deployed. Third, while software modifications to the AP are feasible, the system should not require changes to the 802.11 standard. Fourth, it should be able to assist both static as well as adaptive PSM clients, since both modes are commonly used in clients today. Last, the system should be able to handle multiple PSM clients simultaneously.

5.2 NAPman

NAPman scheduling is designed to minimize energy consumption of PSM devices while still ensuring fairness. In order to be energy efficient, PSM clients that wake up need to receive their packets immediately, say via high priority, so that they can go back to sleep. In order meet the fairness criteria as defined in Section 3.2, packets for PSM clients should not be transmitted preemptively before packets of other clients that arrived earlier at the AP.

Energy-Aware Fairness: NAPman combines energy efficiency with fairness as follows. The AP first checks if it is fair to transmit one or more PSM packets for a given client at the next available opportunity; only if the check passes for a given client does the AP notify the presence of PSM packets for that client. Once a client is notified and the client informs the AP it is ready to receive its packets through a PS-POLL or NULL frame, the AP prepares to transmit the PSM packet(s) using a high priority queue. Before sending the PSM frame, the AP must ensure servicing the PS-POLL message would not result in unfairness. This could happen from an adversarial PSM client or simply when the MAC-layer ACK to the PS-POLL message is lost and the PS-POLL is retransmitted. Pseudo-code of the energy-aware fair delivery scheme used in NAPman for both static and adaptive PSM clients is shown in Figure 9. We detail the pseudo-code of the NAPman energy-aware fair scheduler below.

Static PSM: Handling static PSM clients in NAPman requires only a simple extension to the static PSM implementation available today. A timestamp is attached to each packet on arrival at the AP driver. If the timestamp of the packet at the head of the PSM queue for a client is less than the timestamp of the packet at the head of the main FIFO transmission queue, transmission of the PSM packet at the next available opportunity is fair (lines 45–50). In this case, the PSM queue for this client is advertised as non-empty either in the upcoming beacon frame using the TIM bit (line 5) or in an already outgoing PSM packet for the same client using the MORE bit (line 14). Once a PS-POLL is received, the PSM packet for the respective client is simply transmitted through a high priority queue (line 15) since the previous checks already ensured that such transmission would be fair.

Adaptive PSM: Handling adaptive PSM clients in NAPman requires special attention. This is because existing AP implementations consider adaptive PSM clients as normal CAM clients as soon as they switch from PSM to CAM, and thus do not continue to maintain any PSM-related state for these clients. A major drawback of these schemes is they perform poorly in the presence of heavy background traffic for clients using aggressive idle timeouts, like the iPhone as seen in Section 4.2. Therefore, NAPman continues to maintain state for adaptive PSM clients even after they transition to CAM in order to perform energy-efficient fair scheduling.

On receiving a NULL frame from a client that has switched from PSM to CAM, only those packets that are buffered in the PSM queue and are fair for immediate transmission get enqueued into the high priority queue (lines 18–21). The client receives these packets

```
1  # Called just before beacon transmission for Virtual AP VAP_j
2  UpdateBeaconTIM(VAP_j)
3          Foreach Cl_i ∈ PSMclients associated with VAP_j
4              If isPacketFair(Cl_i)
5                  Set TIM bit for Cl_i in this beacon
6
7  # On receiving a PsPoll frame from Cl_i
8  ReceivePsPoll(Cl_i)
9          # Verify it's fair to send this PSM frame
10         If (!isPacketFair(Cl_i))
11                 return
12         Pkt = Dequeue(PSMQueue(Cl_i))
13         If isPacketFair(Cl_i)
14                 Set MORE bit in Pkt
15         Enqueue(TxQueueHigh, Pkt)
16
17 # On receiving a Null frame from Cl_i waking up
18 ReceiveNullWakeup(Cl_i)
19         while( isPacketFair(Cl_i) )
20                 Pkt = Dequeue(PSMQueue(Cl_i))
21                 Enqueue(TxQueueHigh, Pkt)
22
23 # On receiving a Null frame from Cl_i going to sleep
24 ReceiveNullSleep(Cl_i)
25         Update average IdleTimeout_i for Cl_i
26
27 # Called after a packet to Cl_k was successfully transmitted
28 PacketTxCompleted(Cl_k)
29         If ( Cl_k ∈ AdaptivePSMclients )
30                 CurrentTime = GetTimeOfDay()
31                 SleepTime_k = CurrentTime + IdleTimeout_k
32         Foreach Cl_i ∈ AdaptivePSMclients
33             If ( isPacketFair(Cl_i) && isNotNearingIdleTimeout(Cl_i) )
34                     Pkt = Dequeue(PSMQueue(Cl_i))
35                     Enqueue(TxQueueHigh, Pkt)
36
37 # Check if Cl_i will be awake for at least next delta ms
38 isNotNearingIdleTimeout(Cl_i)
39         CurrentTime = GetTimeOfDay()
40         If ( SleepTime_i - CurrentTime > δ )
41                 return True
42         return False
43
44 # Check if Cl_i has a PSM packet that is fair to transmit
45 isPacketFair(Cl_i)
46         TS_i = Timestamp(Head(PSMQueue(Cl_i)))
47         TS_j = Timestamp(Head(TxQueue))
48         If ( TS_i < TS_j )
49             return True
50         return False
```

Figure 9: NAPman energy-aware fair scheduling algorithm

immediately, and then enters an idle timeout phase. The remaining packets that were not deemed fair for transmission at this time, if any, and any newly arriving packets are buffered in the PSM queue for that client. While the client is idling in CAM, the AP continues to check if any packets in its PSM queue have become fair for transmission. This fairness check is performed anytime a packet is dequeued from the main transmission queue (lines 28–35). In addition to the fairness check, we also need to make sure that the client will not go to sleep in the near future due to a idle timeout (line 33). This is necessary to avoid the transmission of any packets at the tail end of the timeout interval, which might result in unnecessary retransmissions if the client goes to sleep. This check (lines 38–42) requires the AP to know the idle timeout value for each adaptive PSM client. NAPman learns this over time by keeping track of the transition times for each client (lines 25, 29–31).

Finally, CAM clients sometimes also use NULL frames to inform APs to buffer packets for them to perform background channel scanning operation. However, such use of NULL frames is quite infrequent since scanning usually happens in the order of few tens of seconds (the default scanning interval in many Atheros-based Windows cards is 60 seconds). Thus, NAPman doesn't categorize these clients as adaptive PSM clients.

Supporting Multiple PSM Clients: Although the energy-aware fair scheduling ensures background traffic from CAM clients will not adversely impact the energy savings of a PSM client, it does not address the problem of multiple PSM clients competing with each other. When multiple PSM clients are connected to the same AP, all of them wake up at the same time for the beacon frame to check for the presence of any buffered packets at the AP. If multiple clients request for their buffered packets in the same beacon interval, PSM packets destined for different clients get queued in the AP's transmission queue resulting in longer wait times for some of the clients to retrieve the PSM packets. Use of high priority queues also doesn't solve this problem, as we saw in Figure 3. Furthermore, using a shorter beacon interval and advertising the presence of buffered packets at different beacons also doesn't help as we demonstrate in Section 8.1.4.

NAPman leverages virtualization support that is available in most APs today to address this issue. Virtualization enables one physical AP to pose as multiple virtual APs, each having a separate beacon of its own. The basic idea is to have a dedicated virtual AP for each PSM client in order to completely avoid any contention from traffic due to other PSM clients. However, one limitation with virtualization support is that only a limited number of virtual APs can be created with one physical AP. For example, the Atheros chipset allows up to four virtual APs. NAPman controls how clients associate with different virtual APs similar to the approach used in DenseAP [8]. By evenly distributing clients among multiple virtual APs and ensuring that beacon transmissions for the virtual APs are staggered in time, NAPman reduces the number of PSM clients that wake up at the same time. When the maximum number of virtual APs has been reached, then NAPman can assign multiple PSM clients to the same virtual AP using heuristics such as least-loaded, etc.

In summary, NAPman can achieve energy savings comparable to the high priority delivery scheme, without trading off fairness. In addition, NAPman reduces contention between multiple PSM clients by leveraging AP virtualization. One of the key design features is its simplicity. We now describe the implementation of NAPman.

6. IMPLEMENTATION

We prototyped NAPman using the MadWifi v0.9.4 driver [24] for Atheros-based WiFi cards on the Linux platform. We chose MadWifi since it has a stable implementation of AP mode, supports multiple transmission queues with different QoS priorities, and supports AP virtualization. By default, the AP implementation in MadWifi uses normal delivery scheme for PSM traffic. All PSM packets are sent out using the queue for best-effort traffic. We modified the driver to support high priority schemes by inserting the PSM packets in one of the higher priority transmission queues, which gave it priority over best-effort background traffic.

To implement the NAPman fair scheduling algorithm, we modified MadWifi to attach a timestamp in the descriptor associated with each packet when it is handed to the driver. We request an interrupt to be generated after each packet is transmitted, and we update information about the last packet sent in the interrupt service routine. Each PSM packet keeps track of the background packet that is inserted into the driver immediately before it. The fairness check routine compares the timestamp of the most recent packet transmitted with the timestamp of the packet in the head of PSM queue for each PSM client. We modified the NULL frame handler to manage adaptive PSM clients in NAPman. MadWifi by default removes all the packets in the PSM queue for an adaptive PSM client and enqueues them into the transmission queue when the client sends a NULL frame with the power management bit set. In NAPman we continue to maintain the PSM state for these clients to ensure PSM packets are not unfairly transmitted.

MadWifi supports the staggered transmission of beacons for each virtual AP, which is required for NAPman. However, this feature requires hardware support to work. Unfortunately, our WiFi cards did not support this feature. Therefore, for the multiple PSM client experiments, we set the beacon interval to a small value (25ms) and simulated staggered beacons by only advertising the TIM bit for each client at the appropriate beacon corresponding to the associated virtual AP. We then measured the overhead of waking up the clients for beacons corresponding to other virtual APs from our power readings, and subtracted this overhead from our final measurements. Note that all single PSM client experiments did not need this modification, and they run with the default beacon interval of 100ms.

WiFi Client	PSM mode
HP iPAQ hw6945	Static PSM
iPhone 3GS	Adaptive PSM
gPhone HTC Magic	Adaptive PSM
HTC Tilt 8900	Static PSM

Table 3: Smart-phones and PSM modes used in our experiments

7. EVALUATION METHODOLOGY

We present the experimental setup and the methodology used to obtain our results. We analyzed the PSM client behavior on multiple phones listed in Table 3. Unless otherwise noted, all experiments with static PSM are based on the HP iPAQ hw6900 series smart-phone, while adaptive PSM runs are based on the iPhone 3GS. The different APs evaluated in our testbed are listed in Table 1. Unless otherwise noted, we use a Linux PC with modified MadWifi driver that implements normal, high priority and NAPman scheduling as the AP for our experiments.

We measure the power consumed by the phone using an external power monitoring device from Monsoon Solutions [25]. The power monitor tool supplies current to the phone instead of the battery and is able to sample the average current drawn by the device at 5000 Hz. We were unable to connect the power monitor to the iPhone since the battery unit is concealed inside the phone and it was difficult to open the case without damaging the phone. For experiments with the iPhone, we examine the NULL frames to measure the time the WiFi radio stays awake, which is a good indicator of the power consumed by the radio interface. We turned down the backlight on the phone and disabled all other communication interfaces such as 3G and Bluetooth in order to isolate the power consumption of the WiFi interface. For experiments other than the auto-rate experiments, we fixed the PHY data rate of the traffic to 1 Mbps and ensured negligible loss on the wireless links. To generate controlled background traffic in our experiments, we use a Windows XP machine with a Netgear WAG511 WiFi network adapter as a CAM client connected to the AP. Unless otherwise stated, we evaluate all experiments for five minutes in order to obtain our results.

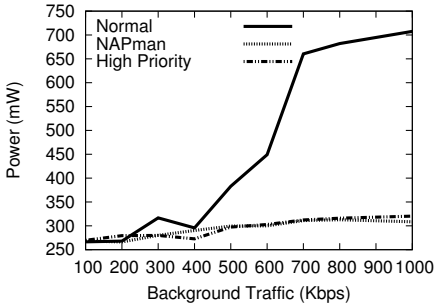

Figure 10: Power drawn by a static PSM client (1 pkt/100ms) with varying CBR background rates

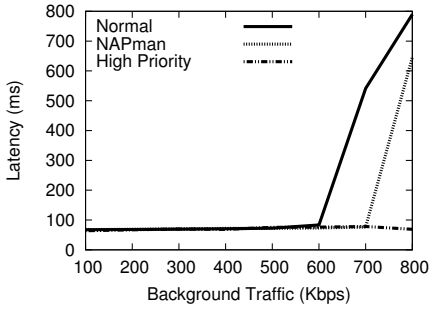

Figure 11: Average latency for packets to a static PSM client (1 pkt/100ms)

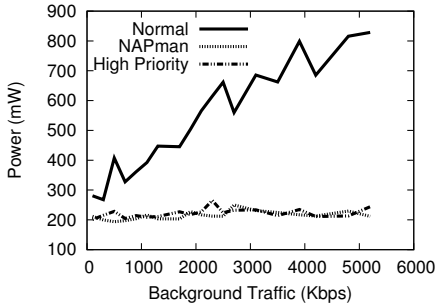

Figure 12: Power drawn by a static PSM client (1 pkt/100ms) with auto-rate

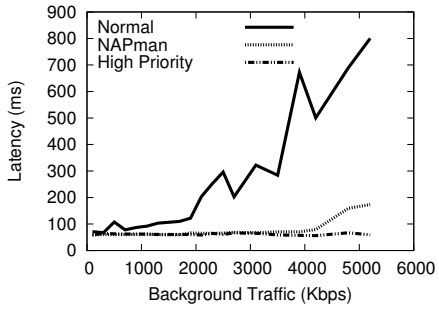

Figure 13: Average latency for packets to a static PSM client (1 pkt/100ms) with auto-rate

8. EVALUATION

In this section, we first start with controlled PSM client traffic in order to highlight various aspects of the NAPman system. We then focus on NAPman performance for various applications such as Internet radio, web browsing, streaming video, IM, etc. We do not evaluate applications that require QoS such as VoIP or interactive gaming since the U-APSD approach of 802.11e (Section 2) is effective for those classes of applications.

8.1 Controlled Experiments

In these experiments the PSM client receives one 1024 byte packet every 100ms. This controlled traffic is intended to represent the worst case for the various scheduling schemes since there is an outstanding packet every beacon interval for the PSM client.

8.1.1 Power Consumption

In Figure 10, we present the power consumption of NAPman's fair scheduling versus normal and high priority scheduling for static PSM clients in the presence of varying background traffic. We can see that the power consumption of NAPman's fair scheduling is similar to the power consumption of the high priority scheduling (the two curves overlap) and results in power reduction of up to 57% compared to the normal scheduling approach. Furthermore, unlike the high degree of unfairness caused by high priority scheduling (Figure 6), NAPman achieves this without causing any unfairness to background flows.

8.1.2 Latency

Figure 11 shows the latency of the various scheduling schemes for a static PSM client. We define latency as the time the PSM packet is sent to the AP driver to the time the PSM packet is confirmed as delivered to the client (timestamped by the ACK received at the AP). At low background rates, the latencies between the scheduling schemes are comparable since the transmission queue does not build up. Once the transmission queue starts to build up (around 700 Kbps), NAPman's fair scheduling actually results in *reduced latency* compared to normal scheduling. This may be surprising if one views fair scheduling as delaying PSM traffic until it is fair in order to advertise the TIM bit (Figure 9), but as we saw in Section 4.3, the unfairness to PSM clients under normal scheduling is in fact rectified by NAPman's fair scheduler. The latency of high priority scheduling stays relatively constant because it can always skip ahead of any pre-existing background traffic.

8.1.3 Impact of Auto-Rate

So far, for simplicity, we used a fixed rate of 1 Mbps for our experiments. In Figure 12, we present the results when the AP is set to MadWifi's default auto-rate algorithm. We move the background client across the lab so the link loss averages around 26% and keep the static PSM client near the AP. The AP is tuned to 802.11b because our PSM client has an 802.11b only card.

As we can see, even with the lowest background rate (100 Kbps), normal scheduling uses 40% more power than both the NAPman and high priority scheduling schemes. With saturated background traffic, normal scheduling uses 3.4 times more power than NAPman.

There are a couple of trends to discuss from this graph. First, even under small background rates, NAPman sees improvement over normal scheduling with auto-rate compared to the fixed rate case where both schemes were identical. This is because the link to the background client can be lossy. The lossiness requires retransmissions which in turn causes build up of the packets in the transmission queue. Second, with normal scheduling, while the

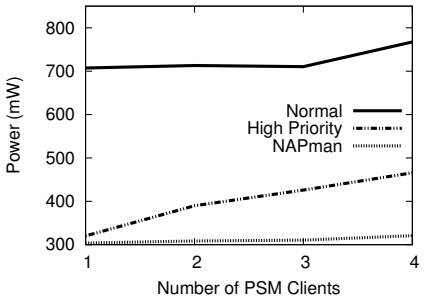

Figure 14: Power drawn by a static PSM client (1 pkt/100ms) with varying number of static PSM clients and saturated CBR background rates

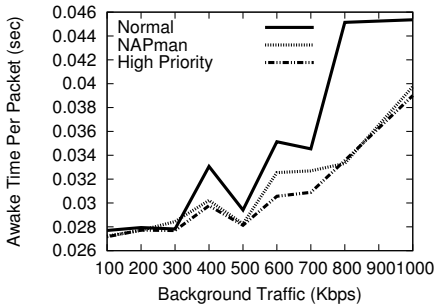

Figure 15: Awake time per packet with varying background traffic for an iPhone 3GS (1 pkt/100ms)

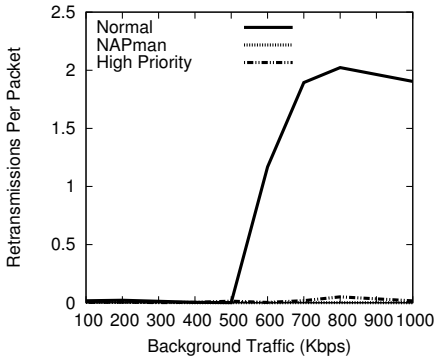

Figure 16: Retransmissions per packet with varying background traffic for an iPhone 3GS (1 pkt/100ms)

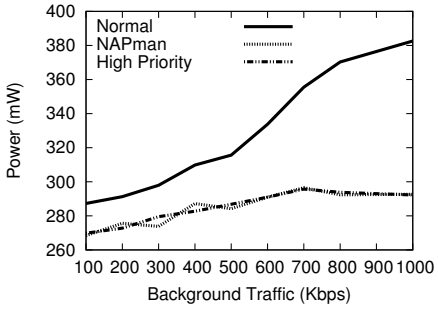

Figure 17: Power drawn by a static PSM client for an eBuddy trace with varying CBR background traffic

power consumed generally rises with increased background traffic, there are significant variations. This is due to variations of the link quality of the background client that affect the actual background usage of the channel. Figure 13 has a similar pattern – as loss fluctuates, so does the latency. We see that the latency of PSM traffic under normal scheduling can be quite high, up to 4.6 times higher than NAPman. This is due to issues with missed PS-POLL opportunities and resulting unnecessary retransmissions as discussed in Section 4.2.

8.1.4 Multiple PSM Clients

In Figure 14 we plot the power consumed when multiple static PSM clients are associated with the AP. We set the PSM traffic to one PSM packet every 100ms and saturated the background traffic with CBR. We vary the number of PSM clients from 2 to 4 and measure the power consumed at one PSM client. Schemes that reduce the amount of time the PSM client waits for its PSM packet (high priority scheduling and NAPman) consume less power than normal scheduling. Furthermore, by reducing contention among PSM clients, NAPman is able to save more power than high priority scheduling. All told, normal scheduling draws 159%-170% more power than NAPman and high priority scheduling draws 43%-64% more power than NAPman. Note that we would expect the high priority and NAPman schemes to eventually converge once the number of clients increases over the number of available virtual APs (we only have 4 smart-phones to test with), however, NAPman would still provide fairness.

Instead of the AP virtualization used in NAPman, one simple solution to isolate PSM clients is to just reduce the beacon interval. This allows the AP to advertise the presence of buffered packets selectively to one or a small number of PSM clients without increasing latency. However, one issue with this simple approach is that all clients will have to wake up for every beacon to see if the beacon has the TIM bit set for it. Our measurements on the iPAQ indicate reducing the beacon interval from 100ms to 25ms results in a 28% increase in overall power consumption of the client. Each client spends a constant amount of time waking up for a beacon (typically a few milliseconds to offset issues with clock skews and to avoid missing beacons). Thus, as the beacon interval is reduced, the amount of time spent awake, and energy consumption, increases super-linearly, making the frequent beacon approach inviable as a power saving solution.

8.1.5 Adaptive PSM Clients

We now consider adaptive PSM clients. In Figure 15, we plot the awake time, per packet, for an iPhone client with demand of one packet every 100ms and varying background rate. In Figure 16 we plot the number of retransmissions for each packet, on average, for the same experiment. We see that NAPman is able to keep similar awake time as high priority scheduling while minimizing retransmissions.

8.2 Trace-Driven Experiments

We now evaluate NAPman using traces from common applications that are used in smart-phones today.

8.2.1 Chat Application

We collected traces for the eBuddy chat application [12] by first running the application and logging and timestamping the packets sent and received. To support repeatability of the experiments,

Trace	Uplink Pkt Size	Uplink Inter-Arrival	Downlink Pkt Size	Downlink Inter-Arrival
128 Kbps Radio	53 bytes	90.89ms	1450 bytes	90.89ms
Pandora	54 bytes	186.32ms	1514 bytes	349.86ms
eBuddy	418.2 bytes	3328.2ms	261.9 bytes	977.05ms

Table 4: Average statistics for trace-driven evaluations

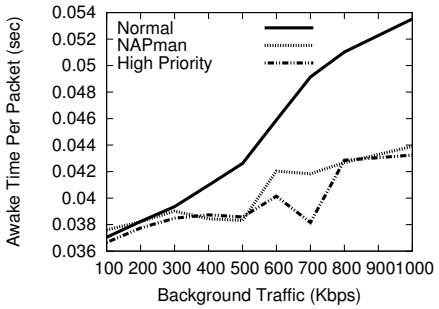

Figure 18: Awake time per packet for a 128 Kbps radio trace with varying CBR background traffic for an iPhone 3GS

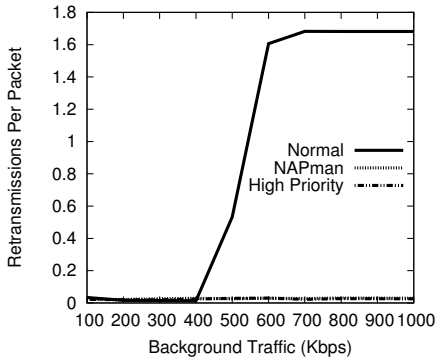

Figure 19: Retransmissions per packet for a 128 Kbps radio trace with varying CBR background traffic for an iPhone 3GS

rather than independently running the application for each experiment, we replay the identically collected trace as traffic to a static PSM client and evaluate the performance of various scheduling schemes. We vary the background CBR traffic rates and plot the power drawn by the PSM client for the different scheduling schemes in Figure 17. Table 4 shows the average traffic statistics for the eBuddy application as well as other applications we evaluate.

We note the following for the eBuddy experiments. First, the base power is higher in these traces because the PSM client must wake up frequently to send updates. The eBuddy traces feature infrequent downlink data, and thus opportunities for improvement are small. Regardless, NAPman delivers up to 30% improvement at saturated background conditions and about 10% improvement for moderate traffic (500 Kbps) over the normal scheduling scheme. The NAPman and the high priority schemes exhibit similar performance.

8.2.2 Streaming Audio

We replay a trace of a 128 Kbps radio stream for an adaptive PSM client. We examine the average per-packet awake time for NAPman, normal scheduling and high priority scheduling in Figure 18. Again, the NAPman approach provides similar benefits to the high priority scheme.

In Figure 19 we plot the number of retransmissions per packet and note similar trends as in Figure 16. The trends for Pandora, another streaming audio application, are similar and are omitted.

8.2.3 Web Browsing

In Figure 20, we evaluate the impact of a web browsing workload for a static PSM client. We use TCP to download a file the size of `www.mobile.msnbc.com`, which is approximately 448 Kb. Each client downloads the page and then sleeps for an average of 39 seconds, which studies have shown is an average viewing time between downloading webpages [11]. We repeat the experiment five times and report the total energy consumed (in Joules) by each of the scheduling schemes. Consumed energy in this case also takes into account the total download time, since download latency with TCP will vary with different scheduling schemes. In the case of normal scheduling, when the background load is high, the connection times out and the download does not even complete (due to the issues discussed in Section 4.2 whereby the client radio goes to sleep while the packet is in the queue). Both NAPman and high priority scheduling use 3X to 4X less energy than normal scheduling with NAPman requiring around 15% more energy than the high priority scheduling under high background load.

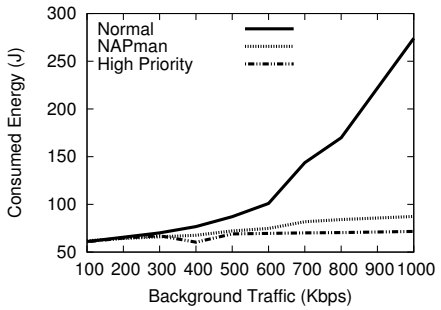

Figure 20: Consumed energy of a static PSM client for a web browsing workload

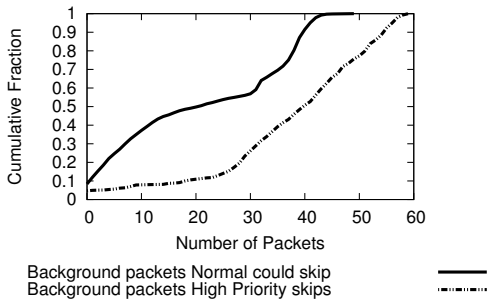

Figure 21: Fairness of a static PSM client for a web browsing workload under saturated background rates

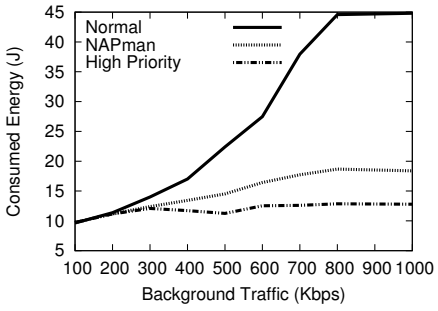

Figure 22: Consumed energy of a static PSM client for a 30 second YouTube workload

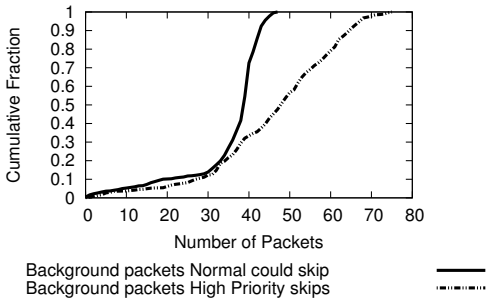

Figure 23: Fairness of a static PSM client for a 30 second YouTube workload under saturated background rates

In Figure 21 we show the fairness results. In the median case about 21 background packets with newer timestamps are scheduled ahead of a PSM packet for the normal scheduling scheme. The unfairness of high priority scheduling is also shown, and in the median case 40 background packets are unfairly skipped by a PSM packet. Note that NAPman scheduling does not unfairly skip any background packets and does not have any background packets with newer timestamps enqueued ahead of it.

8.2.4 YouTube Video

In Figure 22, we emulate a YouTube video download workload. The PSM client uses TCP and downloads a 30 second video encoded at the rate of 330 Kbps. Previous work has shown 330 Kbps to be an average bitrate for YouTube videos [10]. We vary the CBR background rate and plot the consumed energy (in Joules) of each of the scheduling schemes.

Again, we see that normal scheduling performs poorly and the download remains incomplete at saturated background loads. Even under lower loads (300 Kbps), NAPman delivers energy savings of 13% and under a background load of 800 Kbps, NAPman delivers 60% energy savings compared to normal scheduling. NAPman performs worse than high priority scheduling because high priority scheduling can use its unfairness to overtake the median and finish its download faster than the other schemes.

Finally, in Figure 23, we show the fairness results. In the normal scheduling scheme about 38 background packets with newer timestamps are scheduled ahead of a PSM packet in the median case. Again, the unfairness of high priority scheduling is shown and 48 background packets are unfairly skipped by a PSM packet in the median case. As before, NAPman scheduling does not unfairly skip any background packets and does not have any background packets with newer timestamps enqueued ahead of it.

9. NAPMAN ENHANCEMENTS WITH CLIENT SUPPORT

Having described and evaluated NAPman, an AP-based solution, we now step back and ask ourselves what if we had the flexibility to modify the PSM implementation on the clients as well. Could we do anything better than NAPman? In this section we describe NAPman-Dynamic, a variant of NAPman with two major enhancements that utilize client-side support but do not require any changes to the 802.11 standard. The first enhancement deals with how to better manage wakeup schedules of PSM clients, thereby lowering latency. The second one deals with how to put adaptive PSM clients to sleep more efficiently.

Dynamic Beacon Interval: Currently, the PSM client's wake up schedule is tied to the beacon interval. However, if we could inform the client to wake up at arbitrary times instead of the fixed beacon interval, then the AP can fine tune the wake up schedules according to the current traffic demands. When the traffic for a client is high, the beacon interval can be reduced on the fly for the corresponding the virtual AP. This enables the AP to provide lower latencies for PSM clients. NAPman-Dynamic controls when the PSM client wakes up to retrieve the buffered PSM packets. In NAPman-Dynamic, clients are informed about changes in the beacon schedule by using the Beacon Interval field in the beacon frame. Note that existing clients we have examined do not seem to support this option since these clients appear to check this field only during association. NAPman-Dynamic requires simple modifications in the client WiFi driver to keep track of changes in the beacon interval field and appropriately change the wake up schedules.

MORE Bit Indication for Adaptive PSM: The second enhancement addresses the inefficiencies in adaptive PSM clients that arise due to the use of fixed idle timeout intervals before transition to sleep. As summarized in Table 2, aggressive timeouts can result in high packet retransmissions and reduced wireless capacity, while long timeouts can waste too much energy. In NAPman-Dynamic, the AP informs an adaptive PSM client whether there are any more packets available for immediate transmission using the MORE bit. Clients that detect the MORE bit is not set can immediately go to sleep avoiding the need to stay awake for the idle timeout period. Thus, this idea incorporates the benefits of static PSM into adaptive PSM. Existing adaptive PSM clients that we have examined do not look at the MORE bit when they are in CAM mode, but again, this can be fixed quite easily.

NAPman-Dynamic uses the two enhancements described here to manage the sleep and wakeup schedules of multiple PSM clients using a simple slot based reservation scheme. Time is divided into fixed size slots, and the beginning of each slot represents a potential beacon transmission. Thus a slot indicates an opportunity to transmit one or more PSM packets to a client. The conditions to send a PSM packet are similar to NAPman (the packet is fair and sent with high priority), and background packets can be sent in unused slots or during the unused time of a reserved slot. Each virtual AP reserves slots for the clients associated with it. NAPman-Dynamic keeps track of all the slots used by individual virtual APs. When a client first joins the network, the virtual AP reserves the first available slot for its beacon. In addition, the virtual AP also reserves the slots at default beacon interval times for the client, when there are no packets for the client. If a slot is already reserved, it searches for the next available slot and informs the client using beacon interval interval field so that the client knows when to wake up to receive the next beacon. If there are buffered PSM packets for a client at a particular beacon, similar to NAPman, the virtual AP first checks if the packets are fair. If so, it advertises PSM packets for the client

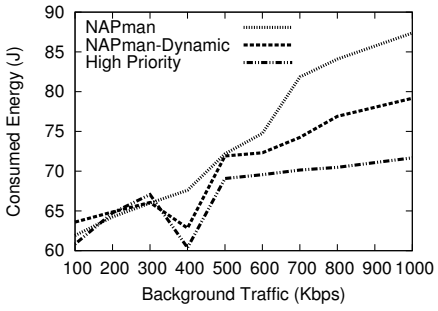

Figure 24: Comparing NAPman-Dynamic with NAPman for a web browsing workload

using the TIM bit. If none of the packets are fair in the current beacon, it estimates the time when the packet(s) becomes fair and reserves the appropriate time slot for transmission.

We implemented NAPman-Dynamic with client support and compare its performance with NAPman and high priority scheduling. We ran a similar workload as in the experiment corresponding to Figure 20. Figure 24 shows the power consumed by a PSM client for the three approaches (since our client did not support adaptive beacon intervals, we manually set the beacon interval to one slot time, 25ms, and subtracted the overhead from waking up for unnecessary beacons). We see that the power consumed by NAPman-Dynamic is lower than NAPman by around 9% at high loads, narrowing the gap with the high priority scheme while still maintaining fairness.

10. CONCLUSION

The energy consumption of a WiFi radio is a significant drain on the battery of mobile smart-phones and thus both APs and smart-phones have implemented a variety of WiFi PSM strategies for saving power. We show that both the normal/high priority scheduling implementations in commercial APs and the static/adaptive PSM strategies used in smart-phones suffer from several serious drawbacks in the presence of competing background traffic, namely, a significant increase of up to 300% in client's energy consumption, decrease in wireless network capacity due to unnecessary retransmissions, and unfairness.

We present the design and implementation of NAPman, an AP-based incrementally deployable solution. NAPman employs a new energy-aware fair scheduling algorithm and leverages virtualization to reduce energy consumption by up to 70% and eliminates unnecessary retransmissions while maintaining fairness. As part of future work, we are looking into extending NAPman to support scheduling algorithms that would allow clients to trade-off latency for energy gains.

11. ACKNOWLEDGMENTS

We thank Venkata Padmanabhan for discussions during the early stages of the project. We appreciate the insightful feedback from our shepherd, Margaret Martonosi, and the anonymous Mobisys 2010 reviewers. We extend our gratitude to Upendra Shevade, who lent his iPhone and provided the iPhone applications to run our experiments.

12. REFERENCES

[1] Y. Agarwal, R. Chandra, A. Wolman, V. Bahl, K. Chin, and R. Gupta. Wireless Wakeups Revisited: Energy Management for VoIP over Wi-Fi Smartphones. In *MobiSys*, 2007.

[2] M. Anand, E. Nightingale, and J. Flinn. Self-Tuning Wireless Network Power Management. In *MobiCom*, 2003.

[3] G. Ananthanarayanan and I. Stoica. Blue-Fi: Enhancing Wi-Fi Performance using Bluetooth Signals. In *MobiSys*, 2009.

[4] T. Armstrong, O. Trescases, C. Amza, and E. Lara. Efficient and Transparent Dynamic Content Updates for Mobile Clients. In *MobiSys*, 2006.

[5] G. Berger-Sabbatel, F. Rousseau, M. Heusse, and A. Duda. Performance Anomaly of 802.11b. In *Proc. of IEEE INFOCOM '2003*, June 2003.

[6] D. Bertozzi, L. Benini, and B. Ricco. Power Aware Network Interface Management for Streaming Multimedia. In *IEEE Wireless Communications and Networking Conference*, 2002.

[7] S. Biswas and S. Datta. Reducing Overhearing Energy in 802.11 Networks by Low-Power Interface Tuning. In *International Conference on Performance, Computing and Communications*, 2004.

[8] R. Chandra, J. Padhye, A. Wolman, and B. Zill. A Location-based Management System for Enterprise Wireless LANs. *NSDI*, 2007.

[9] S. Chandra and A. Vahdat. Application-Specific Network Management for Energy-Aware Streaming of Popular Multimedia Formats. In *USENIX*, 2002.

[10] X. Cheng, C. Dale, and J. Liu. Statistics and Social Network of Youtube Videos. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pages 229–238, 2008.

[11] H.-K. Choi and J. O. Limb. A Behavioral Model of Web Traffic. In *ICNP '99: Proceedings of the Seventh Annual International Conference on Network Protocols*, page 327, Washington, DC, USA, 1999. IEEE Computer Society.

[12] eBuddy Online Messenger. `http://www.ebuddy.com`.

[13] Y. He and R. Yuan. A Novel Scheduled Power Saving Mechanism for 802.11 Wireless LANs. *IEEE Transactions on Mobile Computing*, 8(10):1368–1383, 2009.

[14] Y. He, R. Yuan, X. Ma, and J. Li. Analysis of the Impact of Background Traffic on the Performance of 802.11 Power Saving Mechanism. *Comm. Letters.*, 13(3):164–166, 2009.

[15] Y. He, R. Yuan, X. Ma, J. Li, and C. Wang. Scheduled PSM for Minimizing Energy in Wireless LANs. In *ICNP*, 2007.

[16] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE 802.11, June 1999.

[17] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, IEEE 802.11E, Nov. 2005.

[18] UCSD CSE Wireless Traces. http://www.sysnet.ucsd.edu/wireless/traces/sigcomm2007/.

[19] R. Krashinsky and H. Balakrishnan. Minimizing Energy for Wireless Web Access with Bounded Slowdown. In *MobiCom*, 2002.

[20] S.-W. Kwon and D.-H. Cho. Efficient Power Management Scheme Considering Inter-User QoS in Wireless LAN. In *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pages 1–5, Sept. 2006.

[21] J. Lee, C. Rosenberg, and E. K. P. Chong. Energy Efficient

Schedulers in Wireless Networks: Design and Optimization. *Mob. Netw. Appl.*, 11(3):377–389, 2006.

[22] H.-P. Lin, S.-C. Huang, and R.-H. Jan. A Power-Saving Scheduling for Infrastructure-Mode 802.11 Wireless LANs. *Computer Communications*, 29(17):3483 – 3492, 2006.

[23] J. Liu and L. Zhong. Micro Power Management of Active 802.11 Interfaces. In *MobiSys*, 2008.

[24] Madwifi. `http://madwifi.org`.

[25] Monsoon solutions inc. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[26] X. Perez-Costa and D. Camps-Mur. AU-APSD: Adaptive IEEE 802.11e Unscheduled Automatic Power Save Delivery. In *ICC*, 2006.

[27] D. Qiao and K. Shin. Smart Power-Saving Mode for IEEE 802.11 Wireless LANs. In *Infocom*, 2005.

[28] A. Rahmati and L. Zhong. Context for Wireless: Context-sensitive Energy-efficient Wireless Data Transfer. In *MobiSys*, 2007.

[29] E. Shih, P. Bahl, and M. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *MobiCom*, 2002.

[30] J. A. Stine and G. D. Veciana. A Comprehensive Energy Conservation Solution for Mobile Ad Hoc Networks. In *in: IEEE International Communication Conference*, pages 3341–3345, 2002.

[31] E. Tan, L. Guo, S. Chen, and X. Zhang. PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs. In *ICNP*, 2007.

[32] S. Wong, S. Lu, H. Yang, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *MobiCom*, 2006.

[33] Y. Xie, X. Luo, and R. K. C. Chang. Centralized PSM: an AP-centric Power Saving Mode for 802.11 Infrastructure Networks. In *SARNOFF'09: Proceedings of the 32nd international conference on Sarnoff symposium*, pages 375–379, Piscataway, NJ, USA, 2009. IEEE Press.