Snooze: Energy Management in 802.11n WLANs

Ki-Young Jang* Shuai Hao* Anmol Sheth‡ Ramesh Govindan*
*University of Southern California ‡Technicolor Research
{kjang,shuaihao,ramesh}@usc.edu anmol.sheth@technicolor.com

ABSTRACT

Increasingly, mobile devices equipped with 802.11n interfaces are being used for a wide variety of applications including bandwidth-intensive HD video streaming. Recent work has shown that 802.11n interfaces are power-hungry, so energy management is an important challenge. 802.11n implementations have additional power states relative to earlier generations of 802.11 technology, so energy management challenges for 802.11n are qualitatively different compared to that faced by prior work. In this paper, we describe the design and implementation of Snooze, an energy management technique for 802.11n which uses two novel and inter-dependent mechanisms: client *micro-sleeps* and *antenna configuration management*. In Snooze, the AP monitors traffic on the WLAN and directs client sleep times and durations as well as antenna configurations, without significantly affecting throughput or delay. Snooze achieves 30~85% energy-savings over CAM across workloads ranging from VoIP and video streaming to file downloads and chats.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Design, Experimentation, Performance

Keywords

802.11n, antenna configuration management, energy saving, micro-sleep

1. INTRODUCTION

The capability of mobile devices has increased to the point where a mobile device is often the primary computing device for many people. As such, mobile devices are being used for bandwidth-intensive and/or delay-sensitive applications, such as HD video streaming and VoIP. To meet the demands of such applications, mobile devices are being equipped with the latest high-speed wireless technology, 802.11n.

However, recent work [15] has shown that 802.11n is significantly power-hungry, with power consumption exceeding 2W in some cases. This is a significant concern for mobile devices, whose usability strongly depends upon the lifetime between charges.

Motivated by this finding, we consider the problem of energy management in 802.11n devices. Energy management for 802.11n is qualitatively different from that for its predecessor standards. This is because 802.11n has additional power states: beyond a low-power sleep mode, 802.11n MIMO technology offers the possibility of selectively disabling one or more RF-front ends (*RF-chains*) associated with its antennas, thereby saving energy. These additional power states motivate us to consider two mechanisms for energy management in 802.11n simultaneously: *micro-sleeping* which enables the 802.11n NIC to be put into low-power sleep state for small intervals of time (often a few milliseconds), and *antenna configuration management* which dynamically adapts the number of powered RF-chains. The design of such mechanisms is challenging because they are inter-dependent: changing the antenna configuration affects *airtime* (as defined in [17]) of transmissions, which, in turn, affects the sleep opportunities available for nodes in the WLAN.

Experiments designed to study the potential benefits of these two mechanisms (Section 2) reveal several interesting findings. Both micro-sleep scheduling and antenna configuration management can provide energy gains which vary with the workload type and extent of background traffic to other clients in the network. For low-bandwidth and latency-sensitive traffic like VoIP, exploiting small sleep opportunities during naturally occurring inter-packet gaps can provide a four-fold reduction in energy usage compared to the always-on case. For high-bandwidth traffic like HD video streaming or file downloads more modest gains (a factor of two reduction) are possible by exploiting sleep opportunities at a client when an AP is transmitting to another client, and by selecting the most energy-efficient antenna configuration.

Motivated by these findings, we have designed Snooze (Section 3), a novel energy management system for 802.11n WLANs that has several properties. Snooze *adapts* client sleep durations and antenna configuration to traffic entering and leaving the WLAN, in order to achieve energy-savings; this adaptation creates sleep opportunities by shaping traffic while minimally affecting latency-sensitive applications, and also takes into account the inter-dependence between micro-sleeping and antenna configuration. This adaptation is *AP-directed*, *application-agnostic*, works correctly in the presence of *rate adaptation*, seamlessly incorporates uplink

ACM CoNEXT 2011, December 6–9 2011, Tokyo, Japan.

traffic, and supports multiple *concurrent* heterogeneous applications per client. Finally, Snooze's control messages incur *minimal overhead* by leveraging the frame aggregation capabilities in 802.11n.

In Section 5, we validate these properties using an implementation of Snooze. We show that it can achieve between 30∼85% NIC energy savings over 802.11 CAM (Constantly Awake Mode, in which all antennas are active), over a wide variety of applications ranging from VoIP to HD video streaming, large file downloads and chat. Both micro-sleeping and antenna configuration management contribute significantly to these energy savings, although the proportions they contribute depend on the application. Snooze outperforms 802.11's PSM (Power Save Mode) even for traffic conditions for which PSM was designed, such as that generated by chat sessions. Moreover, Snooze's NIC energy consumption is only 14∼39% higher than that for an ideal energy management scheme in some cases. Snooze's energy savings come at very minimal costs: for file downloads it pays a throughput penalty of 2.5% relative to CAM, and for delay sensitive applications it adds up to 5 ms additional latency on average. Finally, Snooze's control messages consume less than 0.5% airtime and have the property that the control overhead decreases for high-bandwidth applications (where it counts most) because Snooze leverages frame aggregation.

To our knowledge, no other prior work has explored the joint design of micro-sleeping and antenna configuration management in the context of 802.11n WLANs, and prior works on 802.11a/b/g energy management lack at least one of the properties listed above (Section 6).

2. BACKGROUND AND MOTIVATION

In general, Wi-Fi energy management schemes attempt to choose the best *power state* that meets a specific objective, such as performance or energy-efficiency. In 802.11a/b/g implementations, NICs normally have four power states: transmit (Tx), receive (Rx), idle and sleep.

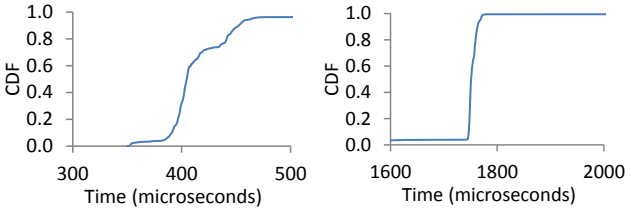
2.1 802.11n Energy Management

Unlike its ancestors, the 802.11n standard defines MIMO transmission and reception capabilities, where multiple antennas can be simultaneously used to increase throughput or provide diversity. Accordingly, 802.11n NICs have an antenna selection capability in which a subset of antennas can be chosen for transmission or reception. The RF-processing components (or *RF-chains*) corresponding to unused antennas can then be powered down. This creates additional device power states, each with different power consumption, as shown in Table 1 [1] for the Intel 5300 3x3 MIMO NIC, and the Atheros AR5BXB92 2x2 MIMO NIC. For these NICs, Idle, Tx and Rx costs vary nonlinearly with the number of RF-chains used. Moreover, sleep mode power consumption is dramatically lower than other modes.

[1]Table for Intel 5300 NIC reproduced with permission from [15]. We measured data for the Atheros NIC using the same technique.

# RF	Intel WiFi Link 5300				Atheros AR5BXB92			
chains	Tx	Rx	Idle	Sleep	Tx	Rx	Idle	Sleep
1	1.28	0.94	0.82	0.10	1.24	0.80	0.72	0.12
2	1.99	1.27	1.13	0.10	2.15	1.16	0.98	0.12
3	2.10	1.60	1.45	0.10	-	-	-	-

Table 1: Power consumption (in Watts) for various modes of Intel (3x3 MIMO) and Atheros (2x2 MIMO) NICs



(a) Sleep operation overhead (b) Wake up operation overhead

Figure 2: Sleep and wake up operations

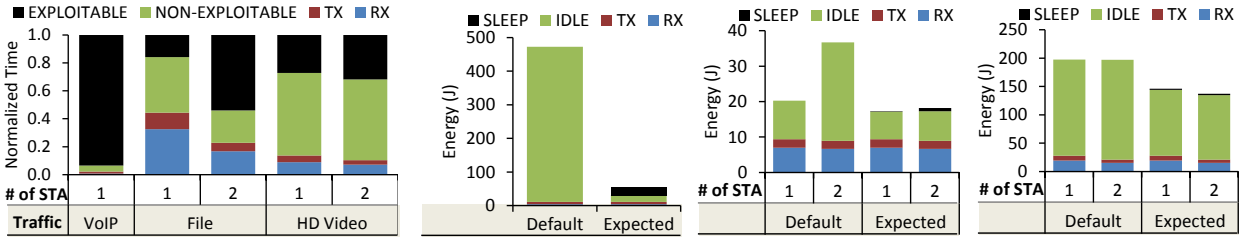
With more power modes than previous 802.11 standards, 802.11n provides an opportunity to design a novel energy management mechanism for clients, which combines *micro-sleep scheduling* (putting the client NIC in sleep mode) with *configuration management* (setting the client's antenna configuration in an energy-efficient state). As motivation, we first experimentally demonstrate that significant energy savings can be obtained by such mechanisms for realistic traffic.

Setup. We conducted a series of experiments using three qualitatively different types of traffic: a delay-sensitive VoIP call from the US to Singapore, a bandwidth-intensive 100 MB file download and a delay-sensitive *and* bandwidth-intensive 140MB 1080p HD streaming video clip whose duration was 2m 17s. Unless otherwise specified, the file and HD traffic was sourced from a server connected by Ethernet to an 802.11n AP. For these two traffic sources we conducted experiments with one and two clients. The VoIP call was placed from an 802.11n client associated with an AP on our campus network. In all cases, we logged all the traffic (including RTS, CTS and other control frames) at two monitors and calculated the energy consumed using the detailed 802.11n energy model presented in [15]. Unless otherwise specified, the default auto-rate algorithm was enabled in each experiment and the driver was free to select the optimal antenna configuration for the channel states during the experiment.

2.2 Micro-sleep Scheduling

In this section we quantify the idealized energy savings achieved by micro-sleep scheduling across different traffic types. In a WLAN, clients can exploit at least two kinds of *sleep opportunities*. First, when an AP is transmitting to client A, another client B can be put to sleep. Second, interframe gaps can be exploited by putting the client to sleep for the duration of the gap. This is, of course, idealized, since it assumes that interframe gaps are known before-hand, and that the client can be put to sleep for arbitrarily small intervals (a capability sometimes called *micro-sleep*).

Micro-sleeping is possible on most 802.11n NICs, by using the same driver functionality used by 802.11 PSM to put NICs in sleep mode. However, there are practical limits to



(a) Normalized time (b) VoIP (c) File downloading (d) HD video streaming

Figure 1: Normalized time and energy consumption for VoIP, File transfer and HD, after applying 2ms filter

micro-sleeping, since the sleep and wakeup operations on modern NICs incur delay and energy overheads. Figure 2 shows the distribution of the times taken, measured by instrumenting the driver, for the sleep and wakeup operations, over 500 invocations of each operation, for the Intel 5300 NIC used in all our experiments. The sleep operation takes between 350 and 500 μs, but with most operations clustered around 400 μs. The time to wakeup the NIC from low-power sleep is, in most cases, a little less than 1800 μs. Thus, the overhead of a sleep, followed by a wakeup, is about 2ms for this NIC. Since we are unable to directly measure the energy cost of sleep and wakeup operations, we use the idle power consumption of the antenna state of the NIC just before going to sleep as an upper bound for these operations.

Figure 1(a) depicts the normalized Tx, Rx and idle times for clients in each of our experiments. (Unless otherwise specified, our plots show *client* airtime and energy consumption, since the focus of our approach is to reduce energy usage at clients). Idle times are categorized as *exploitable* if they are longer than 2 ms and *non-exploitable* otherwise.

VoIP. During our VoIP call, which involved a normal conversation between two participants, a very large proportion of the total call duration was exploitable for micro-sleeping (Figure 1(a)). We also conducted VoIP calls to other destinations (China and New York), and the results were comparable. An examination of the distribution of interframe gaps (also omitted for brevity) which takes both the uplink and downlink traffic into account shows that 70% of the frames have an interframe gap of a few hundred μs and 90% are within 10 ms. Thus, it is clear that the tail of the distribution contributes to the large exploitable idle time, and this tail likely arises from silence suppression as well as delay variability over the wide area network.

Figure 1(b) shows the breakdown of energy attributable to different power states. Without exploiting sleep opportunities, the call would consume 472J. However, if we could perfectly exploit all sleep opportunities, the call would consume 55J, a more than 8-fold reduction in energy usage.

File Download. When a single client downloads the file, less than 16% of the idle time is exploitable (Figure 1(a)). This is because traffic is backlogged on the AP, and almost 99% of the interframe gaps are within 1 ms. Correspondingly, the energy savings by exploiting interframe gaps for a single client is relatively modest (about 15%, Figure 1(c)). However, when two clients download the file simultaneously, the exploitable idle time on average is 54% (Figure 1(a)). This results in a 50% energy savings (Figure 1(c)), more than three times that for a single client. Thus, even for backlogged traffic, moderate energy savings can be obtained; as the number of clients increases, the savings will increase since each client will be presented with more sleep opportunities when the AP is transmitting to one of the clients.

HD Video Streaming. In this scenario, even though the exploitable idle times are higher than the file download scenario (about 27% for the one client case and 32% for the 2-client case, Figure 1(a)), the corresponding energy savings are comparable (26% to 30%, Figure 1(d)). Streaming traffic has naturally occurring exploitable interframe gaps, but uses more antennas for faster download resulting in higher Tx and Rx energy usage and hence more modest energy gains.

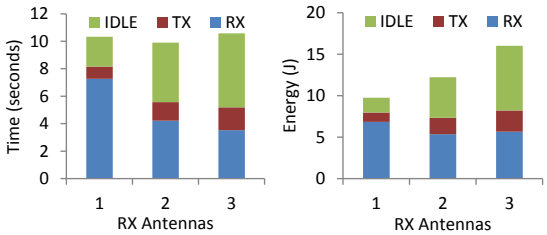
Summary. Thus, depending on the traffic characteristics and the number of clients in the WLAN, upwards of 30% energy gains are possible. However, for some types of traffic much of the gain comes from exploiting sleep opportunities during interframe gaps, while for others it comes from exploiting sleep opportunities when the AP is transmitting to another client. As described above, these are idealized savings: in the next section, we describe practical mechanisms for achieving energy savings through micro-sleep scheduling. Finally, we have not quantified the sleep savings from another commonly used workload, Web browsing. For that workload, user think time dominates the idle time and can result in significant savings, as [23] has demonstrated.

2.3 Antenna Configuration Management

Beyond exploiting sleep opportunities, 802.11n presents a unique energy management opportunity, *antenna configuration management.* Different antenna configurations use different numbers of RF-chains and therefore have significantly different power requirements (Table 1). We now quantify the energy benefits of antenna configuration management and show how it varies based on traffic load and link quality.

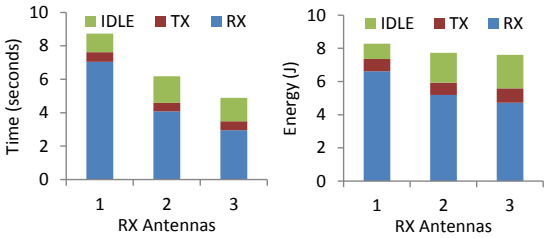
To conduct the experiments below, we explicitly configured clients to use 1, 2 or 3 RF-chains and measured both the time taken to complete the experiment and the energy usage for the experiment. For illustrative purposes, we chose the file download application for the case of a single client, but chose three different settings as discussed below.

Internet as Bottleneck. In this experiment, the wired link



(a) Time (b) Energy Consumption

Figure 3: Internet as bottleneck



(a) Time (b) Energy Consumption

Figure 4: High bandwidth scenario

between the file server and the AP was a bottleneck so the effective data rate required for the transfer was constrained by the wired link. This setup emulates many home and small office settings where the bottleneck resides in the wired or access network. As shown in Figure 3(a), while the download finish times are comparable across different antenna configurations the energy consumption varies significantly (Figure 3(b)). The SISO (single antenna) configuration consumes the least energy (60% lower than three receive antennas) while using a lower effective data rate compared to the multiple antenna configurations.

High Bandwidth. In this experiment, the file server resides at the AP, so that the full wireless bandwidth is available for file download. In such settings, the MIMO capability can result in faster downloads by using multiple antennas (as shown in Figure 4(a), where the completion times for this scenario are faster than the previous scenario), but may incur higher power consumption because more RF-chains are enabled. In our experiment (Figure 4(b)) all three configurations have comparable energy consumption. Interestingly, however, the 3-antenna configuration completes the download about 50% (4 s) faster (Figure 4(a)) than the single antenna configuration. This is the best configuration for this experiment, since it utilizes less airtime overall, freeing up airtime usage for other clients and thereby potentially improving channel utilization.

Low Link-Quality. A similar conclusion can be drawn from an experiment in which the file server was placed at the AP, but the client was moved away from the AP to induce a poor quality link and trigger packet loss. Download times for this setting are more than twice that for the high-bandwidth scenario (results omitted for brevity). With poor link quality, multiple antennas can provide spatial diversity and select the best rate depending on channel characteristics. Here too, the 3-antenna configuration is most efficient since it finishes fastest, even though all configurations have comparable energy consumption.

Summary. These experiments show that energy consumption and channel utilization can be significantly affected by the choice of antenna configurations. Moreover, unlike 802.11a/b/g where the fastest configuration is the most energy efficient, different 802.11n antenna configurations are optimal for different settings and an energy management system must select the appropriate antenna configuration adaptively, based on traffic load and link quality.

3. THE DESIGN OF Snooze

Broadly speaking, there are several architectural designs for energy management in 802.11n networks. Specifically, we are interested in solutions that reduce energy consumption at clients in an application-agnostic manner, using micro-sleeping and antenna configuration, without significantly increasing packet delay. Clearly, there is a continuous tradeoff between energy and delay; however, to exploit this continuum requires application-specific measures, and is beyond the scope of our work. Each client could independently and unilaterally decide when to sleep (sleep times), for how long (sleep durations), and which configuration to use based on purely local information (such as the traffic between the client and the AP) or based on global information (such as the traffic between the AP and other clients) obtained from the AP. Alternatively, the AP can directly coordinate these decisions across different clients, especially when downlink traffic dominates [16].

In Snooze, we explore the AP-directed approach. Our experiments in Section 2 show that energy management decisions depend heavily on the type of workload and also depend on the existence of, and traffic load to other clients in the network. The AP has the most visibility into traffic to and from clients, and is in the best position to make energy management decisions. Our current work focuses on a single AP setting: coordinated energy management across multiple APs is left to future work.

Strawman. To illustrate the challenges underlying the design of Snooze, we start with a strawman design: The AP periodically broadcasts beacons which serve to synchronize all clients and assigns each client a slot in a fixed-width schedule. This approach, resembling time-division multiplexing, requires each client to wake up when it expects to receive the beacon, then go back to sleep, and wake up again just before its time slot.

The approach has several desirable properties. If beacon periods are relatively short, the delay incurred by downlink traffic can be bounded. Moreover, if the interframe gap on traffic to a client is less than the beacon period, traffic is queued at the AP and transmitted when the client awakes, thereby *creating* sleep opportunities. Finally, if the AP observes that traffic to a client requires more airtime than its allocation, it can adjust the antenna configuration to use more

antennas to fit the airtime budget.

Strawman Limitations. However, this strawman design has several drawbacks. If traffic is highly skewed so that one client is downloading streaming video while other clients are browsing the web, this static approach is not *traffic-adaptive* and can add significant delay for the streaming video and miss sleep opportunities for the web browsing clients. Moreover, when the antenna configurations are adapted to minimize energy, the time used by the client may increase or decrease and a static allocation may either miss sleep opportunities or increase delay. Finally, if the beacon interval is kept short to bound delay, the overhead of sleep and wakeup operations, together with the fact that all nodes have to wake up to hear the beacon, can result in very small energy savings.

Snooze leverages the advantages inherent in the strawman, but fixes some of the drawbacks by *jointly adapting* sleep and wake-up durations and antenna configurations to traffic characteristics, and by avoiding the use of broadcast beacons to notify nodes of sleeping opportunities.

3.1 Micro-sleep Scheduling

The goal of Snooze's micro-sleep scheduling component is to determine, for each client i, a sleep and wake-up schedule. From the perspective of a client i, the schedule consists of a sequence of *sleep-wakeup periods*. At any given instant, the current sleep-wakeup period includes a sleep duration S_i, followed by a wake-up duration W_i. The key idea behind Snooze's micro-sleep scheduling algorithm is to determine the S_i based on the packet inter-arrival times, which allows Snooze to exploit sleep opportunities between frames, and to determine the wake-up durations W_i based on the traffic load between the AP and the client.

Determining S_i. Suppose that the AP has just serviced client i and needs to compute the next sleep-frame for that client. The AP maintains a weighted moving average R_i of the packet arrival rate in downstream direction. It computes S_i using:

$$S_i = \min(\max(\frac{1}{R_i}, S_{min}), S_{max})$$

where S_{min} is a configurable minimum sleep time and S_{max} is a configurable maximum sleep time. The lower bound is necessary given the non-trivial costs of the sleep and wake-up operations. The upper bound limits the worst-case latency introduced by sleeping. Moreover, if the instantaneous arrival rate exceeds R_i, packets are queued while the client sleeps; in effect, our approach shapes traffic to create sleep opportunities.

Determining W_i. Ideally, W_i should be proportional to the traffic load. In Snooze, W_i is set to a moving average of the *airtime* used both by uplink and downlink traffic during the previous intervals when the client was awake. Thus, W_i is larger for a client downloading HD video than for a client participating in a chat session. Intuitively, Snooze reserves the channel for a duration W_i. As we shall see below, W_i is a nominal wake-up duration; clients may stay awake for longer or less than this duration depending upon the instantaneous traffic load. Also, W_i depends on the antenna configuration, as described below: if fewer antennas are used in order to conserve energy, this can increase airtime usage and therefore W_i.

Now, given S_i, W_i, and the current time t, Snooze would ideally instruct the client to go to sleep at t and wake-up at $t + S_i$. However, when client i wakes up, another client j might also have been scheduled to wake up, resulting in a *schedule conflict*. Because the AP has global knowledge of all the schedules, it can avoid nominal schedule conflicts by increasing S_i so that i's wake-up period does not conflict with any another client's wake-up period. By contrast, client-directed approaches [19] may not be able to avoid schedule conflicts and can miss sleep opportunities.

Signaling the Client. Having determined these two quantities, Snooze *unicasts* a control message to client i to sleep immediately for a duration S_i. This does not require time synchronization between the AP and client, and assumes that over short durations the clock skews between them are negligible. If this control message is lost, a sleep opportunity is missed but traffic is not adversely affected.

Servicing the Active Client. The AP queues traffic destined for sleeping clients. Snooze allocates W_i airtime to transmit this data. However the actual airtime required may be more or less, depending on changes to the antenna configuration or traffic arrival rate.

Snooze could serve the client for exactly W_i and put it to sleep after that, but this could delay some packets significantly since W_i is a measure of the long-term traffic rate and instantaneous rate can be higher than the long-term average. Because Snooze needs to support latency-sensitive applications like VoIP, it uses a token bucket-based approach which permits some elasticity in the actual wake-up duration. Suppose that there are N clients. On the AP, Snooze maintains one token bucket Q_i for each client i. Every time interval T, Snooze adds $\frac{T}{N}$ airtime credit to each Q_i. However, if a Q_i has already accumulated a predetermined limit L of airtime credits, the excess airtime credit is re-distributed across other token buckets. This approach has the property that the amount of airtime credit associated with each client is proportional to its demand. To see why, consider two clients v and c, where the former is downloading high-bandwidth video and the latter is conducting a chat session. Because chat traffic is relatively infrequent, Q_c quickly accumulates L airtime so all the remaining airtime is now assigned to Q_v.

Now, when the AP needs to serve the i-th client, it uses the accumulated airtime to bound the actual number of packets served to the i-th client. The value L permits clients that have been inactive for a while to receive bursts of traffic, and the load-proportional token assignment allows the AP to accommodate instantaneous rate variations. As a result, W_i may underestimate or overestimate the amount of airtime actually needed to service the i-th client. Both of these can result in lost sleep opportunities or reduced efficiency. If the AP

actually uses less airtime than W_i, then the remaining airtime is lost because it cannot be used for other clients. However, immediately after it has finished servicing the client, the AP recomputes new values for S_i and W_i and instructs the client to go to sleep so that it can save energy.

If the AP uses more airtime than W_i, this might impinge upon another client j's wake-up duration; the AP services both i and j simultaneously, but cannot leverage sleep opportunities for i when it is transmitting to j and vice versa. These problems cannot be avoided, since the AP cannot predict future traffic patterns, but that can be possibly mitigated by better estimators. We have left an exploration of this to future work, but note that even our current design provides considerable energy savings without impacting efficiency noticeably (Section 5).

Occasionally, the AP may have no traffic queued up for the i-th client when that client awakes. This may happen because the traffic load changed and the previous packet inter-arrival time estimates are now stale. If so, it doubles the previous value of S_i (up to S_{max}), re-computes a new value for W_i, and instructs the client to sleep immediately. This allows Snooze to quickly converge on the new value of the packet inter-arrival time.

Uplink Traffic. Snooze seamlessly incorporates uplink traffic. In our current implementation, if a sleeping client has a packet to send, it wakes up and sends the packet, and then goes back to sleep if enough of the sleep duration remains to justify the overhead of the sleep operation. Thus, uplink packets are never delayed, but may contend for the channel with the currently active client being serviced by the AP, so may keep that client awake for longer. However, the airtime taken by uplink traffic is taken into account in calculating the W_is and the token bucket, so that the channel is never oversubscribed. An alternative design strategy might be to delay uplink traffic also until the client is scheduled to wake up; we have left this to future work.

3.2 Antenna Configuration Management

The 802.11n standard provides a novel opportunity to reduce energy consumption: finding an energy-efficient antenna configuration. 802.11n drivers use all available antennas by default, and most rate control algorithms try to find a throughput-optimal or goodput-optimal rate configuration that may use all the available antennas. However, as shown in a recent measurement study [15], the configuration which provides the highest throughput may not be the most energy-efficient. As we have seen (Section 2), the most energy-efficient antenna configuration depends upon the workload and the link quality.

The ideal approach to configuration management would take into account the traffic demands as well as the channel state and attempt to find energy-efficient antenna configurations (i.e., which antennas are selected, and what channel widths and rates are assigned). However, such an approach could require expensive firmware-level changes to commodity NICs. Furthermore, rate and channel-width assignments in the same antenna configuration do not affect the power consumption [15], so such an approach may be overkill.

Instead, Snooze only decides *how many* antennas should be used commensurate with the client's traffic load, leaving it to the rate control algorithm to decide *which* antennas are used and what rates and channel widths are assigned to each antenna. In other words, Snooze limits the NIC's configuration search space, and thereby directly controls the power consumption of the NIC, which depends upon how many antennas (or, more precisely, RF-chains) are enabled. Firmware for the Intel and Atheros cards we have examined allow the driver to explicitly limit the configuration search space.

Snooze uses the *airtime utilization* on a link between a client and an AP to find the number of antennas that would consume the least energy for the given workload and link quality. Airtime utilization is both a function of the traffic demand (workload) and a function of the link quality (on a poor quality link, lower data rate or retransmissions will utilize airtime). For the i-th client, when Snooze computes S_i and W_i (i.e., when it is about to instruct the client to sleep), Snooze computes an airtime utilization U_i, which is the airtime actually consumed in the previous wakeup slot, divided by the airtime assigned to the client for that slot. If U_i is low, either the link quality is high or more antennas have been turned on than necessary for the offered load. Conversely, if U_i is high, link quality is poor or the aggregate rate offered by the existing antenna configuration is insufficient to utilize the assigned airtime.

Snooze uses this intuition in its antenna configuration management algorithm. Initially, the number M of antennas enabled is set to 1; M is a state variable indicating the current power state. In any state, when U_i as measured in the previous sleep-wakeup period is less than a configurable parameter U_{min}, M is decremented (unless it is already 1). Similarly, when U_i as measured in the previous sleep-wakeup period is higher than a configurable parameter U_{max}, M is incremented (unless it is already at the maximum number of antennas supported by the NIC).

Whenever M_i changes, the AP includes its value in a control message sent to the i-th client. When the client receives the message, it instructs its driver to limit the configuration search space for *receiving* downlink traffic to M_i. (The client is free to choose any antenna configuration for *transmitting* upstream, since the AP's receive RF-chains are always turned on). In addition, before communicating with client i, the AP restricts its own rate-control search space to M_i. Essentially, this approach uses recent history of traffic to predict the most energy-efficient configuration for the immediate future.

Putting it all together. Micro-sleep scheduling and antenna configuration management continuously adapt their decisions. A change in i's antenna configuration can change W_i, affecting its sleep schedule and possibly that of other clients. Conversely, if traffic demand drops so that U_i drops, that may trigger a change in antenna configuration. Snooze dampens

this feedback by maintaining a moving average for W_i and applying hysteresis for U_i. We leave a detailed study of this feedback loop to future work.

4. IMPLEMENTATION

In this section we describe several implementation challenges that we overcame in implementing the AP and client support for Snooze. We have implemented Snooze AP and client functionality in the 2.6.35.8 Linux kernel and Intel's open source `iwlagn` driver [7]. Snooze is layered on top of 802.11 QoS, preserves the overall 802.11 design, and interoperates with non-Snooze clients as demonstrated in Section 5.5.

4.1 AP Support

Snooze's per-client traffic queues are implemented in a manner similar to the PSM implementation in the `mac80211` subsystem within the Linux kernel. The airtime credit token buckets and sleep and wake up duration computations are also implemented in `mac80211`. These interact with the driver to access the rate table and to obtain airtime consumption information.

Obtaining Airtime Consumption. Recall that transmitted frames must be charged airtime so that we can account for airtime consumption of clients. Unfortunately, we do not know how much airtime will be actually consumed *before* we transmit frames, since the driver makes this decision dynamically. We calculate expected airtime consumption based on the *current* rate control table and charge the client accordingly. Once a transmission finishes, the driver provides the actual airtime consumed, and Snooze adjusts airtime credit accordingly. For received frames, we calculate the airtime consumption by accounting for all the overhead including the preamble, padding, and all control frames.

Updating the Client Wake-up Time. When Snooze sends a control message to the client, that message may not be transmitted immediately because frames may be queued in the driver before it. However, once the message is transmitted, the driver signals the completion of the transmission, so Snooze now has a correct estimate of when that message was actually delivered. It accordingly updates its calculation of when the client will wake up. We need to do this because our control messages indicate sleep durations and not absolute times.

Reducing Control Traffic Overhead. As we have discussed, Snooze unicasts a short control message (whose total frame size is 36 bytes) to each client to specify its sleep duration and its antenna configuration. To reduce overhead, Snooze encapsulates its control messages in an 802.11 QoS *data frame* which allows the driver to aggregate the control message with other data traffic. Snooze's control messages can incur three types of overhead. In low traffic scenarios, control messages may be sent as separate frames and incur the airtime cost of contending for the channel and transmitting the preamble. In higher traffic scenarios, control messages may be aggregated, using 802.11n's built-in support for frame aggregation. In this case, the airtime cost of contending for the channel and transmitting the preamble are amortized over the aggregated frames, but the airtime incurred by the control message must be paid. In some cases, even this cost may be avoided, incurring zero overhead: the control message may fit in the space that might otherwise have been occupied by padding bytes in the aggregated frame [15]. We quantify these different types of overhead in Section 5.5.

		Delay/Jitter Sensitivity	
		High	**Low**
Bandwidth Requirement	**High**	HD video streaming	File downloading
	Low	VoIP	Chat

Table 2: Applications used for our evaluation

4.2 Client Implementation

To the client-side implementation, we added the ability to sleep and wakeup the NIC and set the receive antenna configuration as directed by the control message from AP.

Our testing revealed an interesting interaction while handling frame loss in aggregated frames. In the AP, the Snooze control message is always enqueued at the driver as the last frame during an awake period. In our initial implementation, the Snooze client assumed that it would be safe to sleep immediately upon receiving the control message. There is one case where this is not a safe assumption. Suppose that the Snooze control message is delivered as a sub-frame in an aggregated frame, and that the control message was successfully decoded, but one or more other sub-frames were lost. If the client were to go to sleep, the AP would retransmit the lost frame(s) until the maximum retransmission limit, and would not service other clients. To detect this condition, the client checks sequence numbers when receiving frames. If it finds any frame loss in an aggregated frame which includes a Snooze control message, it conservatively skips the sleep opportunity and stays awake.

5. EVALUATION

In this section, we first describe our experimental methodology, then validate Snooze functionality and evaluate its performance over a variety of traffic workloads. We also demonstrate Snooze's incremental deployability, and quantify its overhead.

5.1 Methodology

In our experiments, we consider four applications with different bandwidth requirements and delay sensitivities (Table 2). For streaming video, we use `VLC` [9] (with RTP) and the same 1080p HD video file used in Section 2. For file downloading, we use `Iperf` [5] to download a 100MB file. For VoIP, we use the VoIP trace described in Section 2, and for chat we collected a 5-minute trace from an online chat room. For the last two applications, we replay the traffic traces using `tcpreplay` [8] to enable a meaningful comparison across the alternatives we study.

Our experiments use seven mini PCs, each of which is a Zotac ZBOX with an Intel Atom D510 processor equipped with 2GB RAM and an Intel Wifi Link 5300 `802.11abgn`

T	L	S_{min}	S_{max}	U_{min}	U_{max}
1ms	5ms	5ms	100ms	.3	.7

Table 3: Configurable parameters and their default values

NIC. Our experimental setup mimics a home/office network. We use a wired connection between AP and a server, and use `dummynet` [2] to configure the bandwidth limit with 25Mbps for download and 10Mbps for upload; these numbers are motivated by residential Internet bandwidths in developed countries reported by [1]. For all our experiments, we use `hostapd` [3] for AP, and we run up to four clients simultaneously. We keep the AP auto-rate control algorithm for all our experiments, since that is the default setting for commercial 802.11n devices.

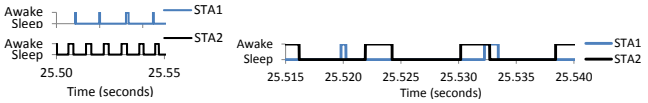
Table 3 shows the *configurable* parameters in Snooze and their default settings in our experiments. We obtained these settings after exploring the parameter space; we have omitted a discussion of parameter sensitivity for space reasons. Two monitor nodes passively collect traffic traces which are then merged to ensure that we capture all transmissions in the network including retransmissions [2]. Each experiment was run 3-4 times with essentially the same results, indicating that losses due to collision do not affect our results significantly.

Calculating Energy Consumption. To attribute the power consumption of the NIC to the different operating states, we do not measure the power consumption directly. Instead, we use the captured traces and the detailed 802.11n NIC energy model presented in [15] to calculate the energy usage for various applications across the different states of the NIC. 802.11n cards send NULL frames before they go to sleep and after they wake up, so from our traces we are able to accurately estimate the sleep time of the NIC. We confirmed that the gaps between sleep times measured by our passive monitoring approach and those reported by a sleep duration register available on our NIC are within 1%. The active transmit, receive and idle times can directly be measured from the network traces. This allows us to concurrently measure energy consumption at multiple clients, unlike some prior work [14, 23, 19].

Comparison and Metrics. We compare Snooze with CAM and adaptive PSM[3] for several metrics measured for each client: total energy usage of the NIC, throughput and delay. Unlike static PSM, which has been shown to incur large delays for interactive applications [21], modern WLAN NICs implement an adaptive PSM technique where the NIC remains in active mode for a short additional timeout interval in anticipation of further traffic to or from the AP. These timeouts vary for different NICs and driver implementations.

[2]We emphasize the Snooze's design does not require passive monitors. We use these only for evaluation.

[3]PSM is the default energy management on almost all commodity NICs. The other energy management techniques, such as U-APSD, S-APSD and SMPS, are not available on every WLAN NIC. The Intel NIC we use only supports PSM.



(a) 1Mbps, 20Mbps flow (b) Overlapping

Figure 5: Sleep and awake in Snooze

We use the default 500ms timeout available on the Intel Linux drivers.

5.2 Traffic Adaptation in Snooze

To visually depict how Snooze adapts to traffic, we perform a simple experiment in which we run two UDP flows of 1Mbps and 20Mbps from the server to different clients. As Figure 5(a) shows, Snooze schedules their wake-up durations relatively disjoint and proportional to their traffic demands. Occasionally, wake-up durations may overlap (Section 3); Figure 5(b) shows one such occurrence in our trace.

5.3 Homogeneous Traffic

In this section, we evaluate Snooze using traffic scenarios in which multiple clients on a WLAN concurrently run the same application.

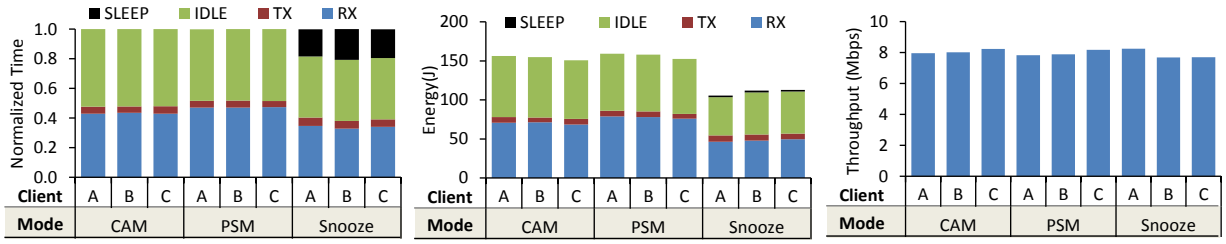
File Downloading. We configure three clients (A,B,C) to each concurrently download a 100MB file from the server. Figure 6(a)[4] shows the normalized time breakdown of each operation (Tx, Rx, idle and sleep), for Snooze, CAM and PSM. Surprisingly, we find that PSM clients cannot sleep and consume the same energy as CAM. This is because adaptive PSM's timeouts are not traffic-adaptive and clients are forced to remain awake when the AP is servicing another client. Other PSM-based approaches [23, 25] may exhibit the same behavior for backlogged traffic. By contrast, all three Snooze clients sleep about 20% of the time on average, resulting in about 30% energy savings, compared with both CAM and PSM, as shown in Figure 6(b).

As Figure 6(c) shows, Snooze does not incur any noticeable throughput drop compared to CAM. The average throughput of Snooze clients is 2.5% lower than CAM and 1.2% lower than PSM. This illustrates the ability of Snooze to leverage naturally occurring sleep opportunities and minimally shape traffic in order to achieve energy savings without sacrificing performance.

HD Video Streaming. We next configure two clients to simultaneously receive 1080p HD video streams, to illustrate Snooze's ability to satisfy the demands of bandwidth intensive and delay-sensitive applications. With `H.264/AVC` and `AAC` encoding, each stream requires about 8∼10 Mbps. As Figure 7(a) shows, PSM clients are unable to sleep in this case, while both Snooze clients sleep more than 25% on average. Snooze provides more than a 2-fold energy reduction, relative to both CAM and PSM as shown in Figure 7(b).

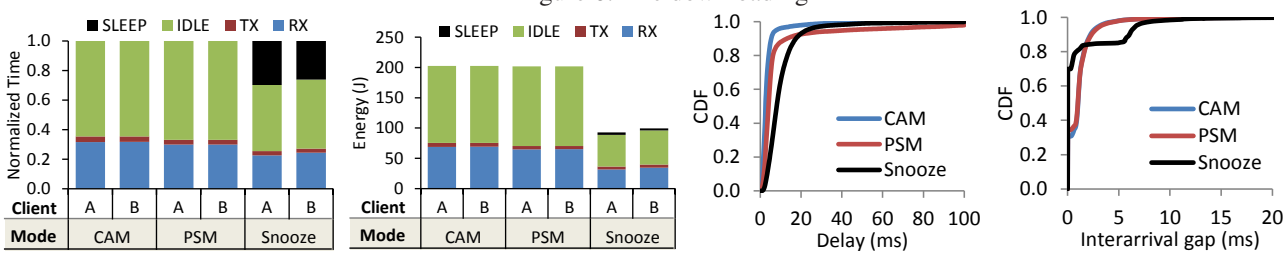
Figure 7(c) shows the distribution of *delay* for each frame, defined as the gap between the time a frame is enqueued to `mac80211` and the time it is acknowledged. Snooze's de-

[4]The small difference between CAM and PSM comes from experimental variation.



(a) Normalized time (b) Energy consumption (c) Throughput

Figure 6: File downloading



(a) Normalized time (b) Energy consumption (c) Delay (d) Packet interarrival gaps

Figure 7: 1080p High Definition video streaming

lay distribution is slightly different from that of CAM and PSM, but the average delay for CAM and PSM are 2.5ms and 4ms respectively, while for Snooze it is around 8ms. Since Snooze shapes traffic to create more sleeping opportunities, it slightly increases delay, compared with CAM and PSM. By trading off small additional delay, Snooze is able to achieve significant energy savings.

Figure 7(d) shows the packet inter-arrival time distribution as observed at the clients. Packets that are aggregated into a frame are assigned an inter-arrival time of zero. Snooze, by shaping traffic at the AP and creating sleep opportunities, is able to leverage frame aggregation more aggressively than PSM or CAM; 70% of the inter-arrival times in Snooze have a value 0, compared to 30% for PSM and CAM. Moreover, greater than 90% of inter-arrival times in Snooze are within 6ms. These small gaps can be smoothed out by buffering at the endpoints, and the increase in device lifetimes resulting from energy savings makes the trade-off worthwhile.

VoIP. Unlike the previous experiments, both end-points in the VoIP call generate low bandwidth, delay sensitive traffic of comparable upstream and downstream volumes. We use the same trace described in Section 2, and we replay the trace on the server and three clients for two minutes. As shown in Figure 8(a), Snooze clients sleep about 60% on average, while PSM clients just sleep about 3%. Overall, Snooze clients achieve a 4-fold reduction in energy usage compared with CAM and PSM.

To achieve these significant savings, Snooze sacrifices about 4ms delay on average as shown in Figure 8(c). This is negligible considering that the ITU recommends an end-to-end delay of less than 150ms for VoIP [6]. Also, the packet inter-arrival time distribution is comparable for Snooze, CAM and PSM, as shown in Figure 8(d).

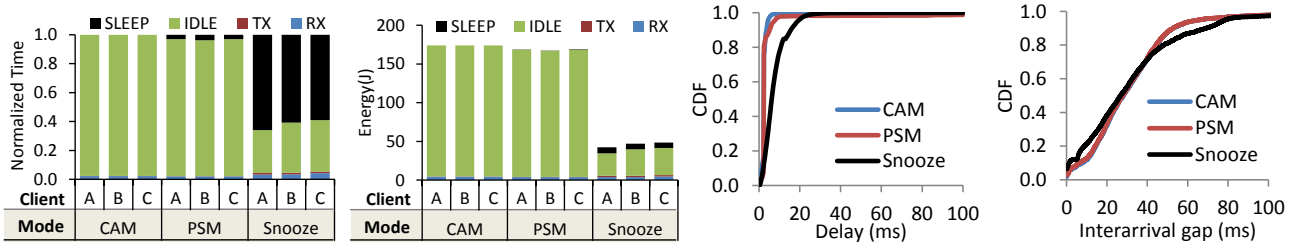
Chat. Chat traffic is *bi-directional* but requires low bandwidth and is delay-insensitive. For this experiment, we replay the chat trace for 2 minutes on the server and 3 clients.

User think times during a chat session present exploitable sleep opportunities. For this reason, PSM clients sleep more than 60% of the time as shown in Figure 9(a). Snooze clients, however, are able to more aggressively exploit sleep opportunities, and sleep more than 85%. This difference arises because, after receiving a packet, PSM clients continue to stay awake for up to several hundred milliseconds in anticipation of additional traffic, while Snooze clients are directed by the AP to sleep immediately. Overall, Snooze obtains a 8-fold lower energy consumption than CAM, and a 3-fold lower energy consumption than PSM (Figure 9(b)).

Finally, PSM can induce delays of 100 ms (the beacon interval) or more, as shown in Figure 9(c), where nearly 40% of traffic is delayed by at least 100 ms. By contrast, without having to customize Snooze parameters for this class of traffic, Snooze introduces only more modest delays; these delays are still larger than what we have seen for other applications, and can be attributed to variability in inter-packet gaps in chat applications, which affects Snooze's determination of sleep and wakeup schedules (Section 3).

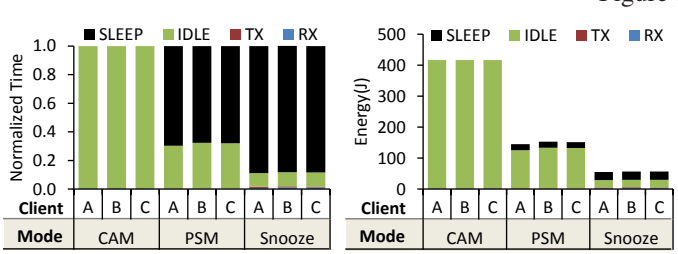
5.4 Heterogeneous Traffic

One Application per Client. To emulate a more realistic setting, we conduct an experiment with a heterogeneous application mix in which we set up four clients, each running a different application: file downloading, HD video streaming, VoIP and Chat. The file download completes in 1 minute while the other applications run for 2 minutes. As Figure 11(a) shows, Snooze can accommodate multiple concurrent applications. More important, even with this heterogeneous application mix, it can leverage sleep opportunities correctly for different applications. As a result, the energy savings obtained by each application are comparable



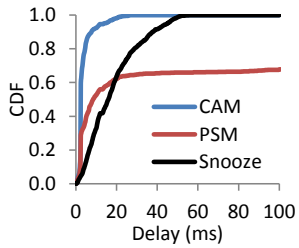
(a) Normalized time (b) Energy consumption (c) Delay (d) Packet interarrival gaps

Figure 8: VoIP



(a) Normalized time (b) Energy consumption

Figure 9: Chat



(c) Delay

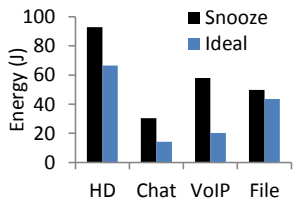


Figure 10: Snooze vs. Ideal

	File	VoIP	HD Video	Chat
CAM	90.35	177.40	175.68	176.48
MS-Only	75.44	81.65	118.60	45.51
Snooze	49.76	57.90	92.90	30.45

Table 4: Deconstructing Snooze energy savings (in Joules). "MS-Only" means only applying micro-sleep mechanism.

to the homogeneous workloads described above: compared to CAM, Snooze saves 45% energy for file downloads, 47% for HD video, 67% for VoIP and 83% for chat.

Deconstructing Snooze Energy Savings. Snooze concurrently uses two inter-related energy-saving techniques: micro-sleep scheduling and antenna configuration management. Understanding the contribution of each technique to the overall energy saving is a little tricky because the mechanisms are not independent: antenna configuration management can alter the sleep opportunities by changing client airtime usage, and micro-sleep scheduling can affect antenna configuration management by queuing packets and increasing utilization when the client is awake. To study this, we ran one additional experiment, using the one application per client heterogeneous mix, but with configuration management in Snooze disabled. Table 4 shows the breakdown of energy contributions. Both techniques contribute significantly to energy savings, but their contribution varies across applications. For example, for file downloads, micro-sleeping contributes 15J in energy-savings and antenna-management about 25J. In contrast, for chat, micro-sleeping contributes 130J and antenna-management only 15J.

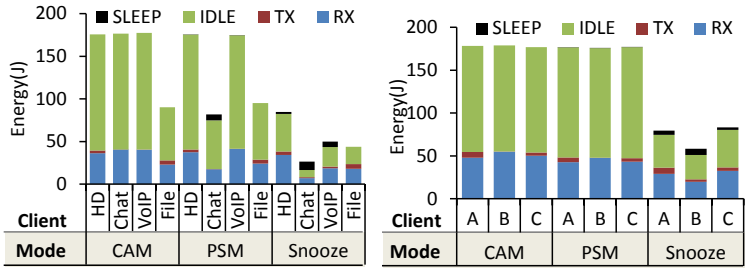
An Ideal Energy Management Scheme. We can also use the previous experiment to quantify how Snooze compares with an ideal energy management scheme. We define the energy usage of this ideal scheme as follows: for each application, we take its CAM trace and compute the total energy that would have been used if (a) every inter-frame gap of more than 2ms could be exploited perfectly, and (b) each transmission only used as much energy as if it were transmitted using only one antenna (regardless of how many were actually used). This ideal scheme will result in an empirically optimal energy consumption without shaping traffic, and assumes zero mode switching costs. For this reason, this ideal is clearly unattainable: indeed, Snooze uses almost 3 times as much energy as the ideal for VoIP and chat (Figure 10). This is unavoidable, because the ideal scheme knows the exact inter-packet gap, but Snooze must expend energy to find out how long it should sleep. More interesting, however, is that Snooze is within 14% and 39% of the ideal for file downloads and HD video, because in these cases the inter-packet gaps are smaller and less variable.

Multiple Heterogeneous Applications per Client. Some mobile OSs support concurrent applications of different types (e.g., chat and file downloads). Unlike some prior work [19, 23, 25], we demonstrate that Snooze can support multiple heterogeneous applications per client. We configured three clients to concurrently run the following applications: client *A* runs VoIP and file download, client *B* runs VoIP and chat, and client *C* downloads HD video. Snooze is able to obtain 3-4 fold energy savings relative to CAM and PSM (Figure 11(b)). Although client *B* runs the chat application, PSM is unable to achieve any energy savings because *B* also concurrently runs VoIP. With Snooze, client *B* is able to achieve almost 4-fold reduction in energy usage relative to PSM.

5.5 Interoperability and Overhead

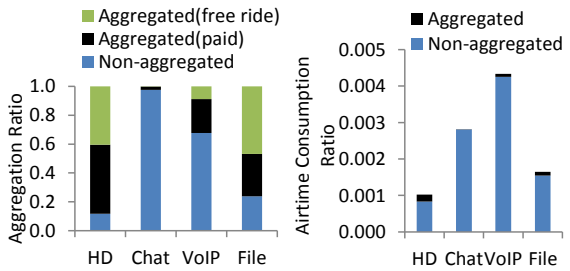
Co-existence with CAM Clients. Since it preserves 802.11 design, Snooze co-exists with non-Snooze clients. To demonstrate this, we set up *A* and *B* as CAM clients, and only *C* as a Snooze client. All clients download a 100MB file.

Unlike PSM, which has been shown to be unfair to CAM clients [23], Snooze achieves comparable throughput and de-



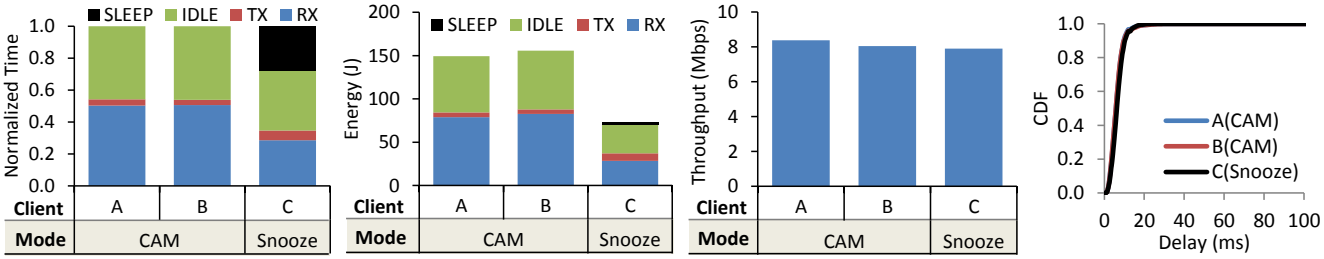
(a) One application per client (b) Multiple applications per client

Figure 11: Heterogeneous traffic energy consumption



(a) Aggregation ratio (b) Airtime overhead

Figure 12: Overhead



(a) Normalized time (b) Energy consumption (c) Throughput (d) Delay in AP

Figure 13: Interoperability

lay to CAM clients (Figures 13(c) and 13(d)). Despite this, Snooze achieves a 2-fold reduction in energy compared to CAM (Figures 13(a) and 13(b)).

Finally, this experiment illustrates an important feature of Snooze also visible in previous results. Because the Snooze client can sleep when the AP is transmitting to the CAM clients, it incurs not just lower idle energy consumption, but also lower *receive* energy consumption than the CAM clients. When the AP transmits to a client, other clients within range incur an energy cost equivalent to a receive operation, even if they are not in monitor mode (we have experimentally verified this for Intel and Atheros 802.11n cards, using similar setup in [15]). Because the Snooze client sleeps when the CAM clients are transmitting, it does not incur this cost.

Low Overhead. We quantify Snooze overhead in two ways for the one application per client heterogeneous mix. To calculate the overhead, we take preamble and ACK overhead into account. As discussed in Section 3, Snooze is designed to take advantage of frame aggregation. Figure 12(a) shows the fraction of Snooze messages that are aggregated or not aggregated. This varies by application: traffic-intensive applications provide greater opportunities for aggregation. Moreover, in many cases, the control messages are essentially free and incur no overhead because they fit within space that would have been occupied by padding in an aggregated frame. For example, for HD video streaming and file downloading, more than 70% of Snooze messages are transmitted as a sub-frame in an aggregated frame. In both cases, more than 40% of Snooze messages are transmitted for free.

Figure 12(b) shows the fraction of airtime consumed by Snooze control messages. For all four cases, Snooze message overhead is less than 0.5%. For high-bandwidth applications, like video and file downloads, Snooze has the nice property that it incurs *less* airtime (0.2% or less), because its design leverages frame aggregation.

		µPM	C-PSM	Catnap	NAPman	Snooze
AP-directed			√	√	√	√
Traffic types	HB-DS					√
	HB-DI	√		√		√
	LB-DS	√			√	√
	LB-DI	√		√	√	√
802.11n support						√
Support for rate adaptation					√	√
Support for multiple heterogeneous apps per STA						√

Table 5: Snooze and other related 802.11 energy management systems. HB: High Bandwidth, LB: Low Bandwidth, DS: Delay Sensitive, DI: Delay In-sensitive.

6. RELATED WORK

While power management in 802.11 has been an active area of research, to our knowledge Snooze is the first work on 802.11n energy management that considers both sleep scheduling and antenna configuration management. Table 5 contrasts Snooze with prior research efforts along different dimensions. In the table, an absence of a check mark indicates that the corresponding dimension was either not discussed or not explicitly demonstrated.

AP-mediation. Existing approaches for AP-mediated power management perform centralized scheduling based on PSM. However, these approaches are not application-agnostic and either make use of application-level proxies to shape traffic [12, 13, 14], are limited to certain traffic types [18, 22, 23, 24], or require explicit hints from the applications [11]. Since these approaches are designed on top of PSM, they may also not be suitable for latency-sensitive traffic like VoIP, and so do not provide the same range of functionality as

Snooze. Non-AP-mediated techniques, like μPM [19], use micro-sleeps for a very short time period like 100μs, but may not be implementable on existing commercial NICs because of the large time overhead involved in switching between the sleep and wake power states of the NIC (Section 2). Moreover, because the AP is unaware of micro-sleeps, they may cause unnecessary retransmissions (and thereby confound the rate adaptation algorithms) leading to severe performance degradation. SleepWell [20] addresses the problem of reducing the power wasted due to increased backoff in an interference-limited dense WiFi deployment, which is orthogonal to the problem being addressed by Snooze. Additionally, SleepWell does not adapt antenna configurations for 802.11n.

Heterogeneous Traffic Classes. Table 5 shows that Snooze provides energy savings across more classes of traffic than prior work. Moreover, Snooze allows multiple concurrent heterogeneous traffic flows per client, a feature not demonstrated in prior work.

802.11n and Rate Adaptation. Unlike any other prior work, Snooze exploits energy savings from 802.11n antenna configuration management, in addition to exploiting micro-sleep opportunities, while achieving low overhead by exploiting frame aggregation. As such, Snooze is unaffected by rate adaptation. In contrast, all existing Wi-Fi power management techniques other than [23] do not support 802.11 rate adaptation and assume that all clients in the network use the same fixed data rate.

802.11 Standard. Finally, the 802.11 standard has defined a variety of power management mechanisms that span the design space of AP mediated approaches like S-APSD [21] and PSM [10], client-driven approaches like U-APSD [21], and approaches targeted for 802.11n like SMPS [4]. Unfortunately, most of these mechanisms do not provide sufficient details and are either not implemented by commercial NICs or are unable to support different traffic types.

7. CONCLUSION

In this paper, we have presented Snooze, an energy management scheme for 802.11n WLANs. Unlike previous work on energy management for WLANs, Snooze combines two novel and related mechanisms: client *micro-sleeps* and *antenna configuration management*. The former enables the AP to coordinate client sleep patterns to maximize energy savings, while the latter dynamically adapts the number of active RF-chains to adapt to traffic bursts and link quality changes. We have implemented Snooze on commercial hardware and have evaluated over several types of traffic including file downloading, VoIP, chat and HD video streaming. Our results demonstrate that Snooze achieves 30$\sim$85% of energy across a wide range of traffic scenarios, while incurring less than 0.5% airtime overhead.

There are a number of open problems related to the design of Snooze: exploring sleep scheduling and antenna configuration management in a multi-AP setting, where explicit coordination may be required between APs; characterizing Snooze performance under highly bursty workloads; studying the impact of Snooze's traffic shaping on upper-layer protocols and application-layer quality metrics; analytically modeling Snooze; exploring the interaction between Snooze and transmit power control; and understanding Snooze's sensitivity to choices of parameters.

8. REFERENCES

[1] akamai. `http://www.akamai.com/stateoftheinternet`.
[2] dummynet. `http://info.iet.unipi.it/~luigi/dummynet`.
[3] hostapd. `http://hostap.epitest.fi/hostapd`.
[4] IEEE 802.11n-2009Amendment 5: Enhancements for Higher Throughput. IEEE-SA. 29 October 2009.
[5] Iperf. `http://sourceforge.net/projects/iperf`.
[6] ITU-T Recommendation G.114, 2003.
[7] iwlwifi. `http://intellinuxwireless.org`.
[8] tcpreplay. `http://tcpreplay.synfin.net`.
[9] vlc. `http://www.videolan.org/vlc`.
[10] 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standards*, 2007.
[11] M. Anand, E. B. Nightingale, and J. Flinn. Self-tuning wireless network power management. In *Proc. of ACM MobiCom*, 2003.
[12] T. Armstrong, O. Trescases, C. Amza, and E. de Lara. Efficient and transparent dynamic content updates for mobile clients. In *Proc. of ACM MobiSys*, 2006.
[13] S. Chandra and A. Vahdat. Application-specific network management for energy-aware streaming of popular multimedia formats. In *Proc. of USENIX ATC*, 2002.
[14] F. R. Dogar, P. Steenkiste, and K. Papagiannaki. Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proc. of ACM MobiSys*, 2010.
[15] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall. Demystifying 802.11n power consumption. In *Proc. of ACM HotPower*, 2010.
[16] K.-Y. Jang, M. Carrera, K. Psounis, and R. Govindan. Passive On-Line In-Band Interference Inference in Centralized WLANs. Technical Report 916, Univ. of Southern California, July 2010.
[17] K.-Y. Jang, K. Psounis, and R. Govindan. Simple Yet Efficient, Transparent Airtime Allocation for TCP in Wireless Mesh Networks. In *Proc. of ACM CoNEXT*, 2010.
[18] R. Krashinsky and H. Balakrishnan. Minimizing Energy for Wireless Web Access Using Bounded Slowdown. In *Proc. of ACM MobiCom*, 2002.
[19] J. Liu and L. Zhong. Micro power management of active 802.11 interfaces. In *Proc. of ACM MobiSys*, 2008.
[20] J. Manweiler and R. R. Choudhury. Avoiding the rush hours: Wifi energy management via traffic isolation. In *Proc. of ACM MobiSys*, 2011.
[21] X. Pe andrez Costa and D. Camps-Mur. IEEE 802.11e QoS and power saving features overview and analysis of combined performance. *Wireless Comm, IEEE*.
[22] D. Qiao and K. Shin. Smart power-saving mode for IEEE 802.11 wireless LANs. In *Proc. of IEEE INFOCOM*, 2005.
[23] E. Rozner, V. Navda, R. Ramjee, and S. K. Rayanchu. NAPman: network-assisted power management for wifi devices. In *Proc. of ACM MobiSys*, 2010.
[24] E. Tan, L. Guo, S. Chen, and X. Zhang. PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs. In *Proc. of IEEE ICNP*, 2007.
[25] Y. Xie, X. Luo, and R. Chang. Centralized PSM: An AP-centric power saving Mode for 802.11 infrastructure networks. In *SARNOFF*, 2009.