to appear in IEEE Workshop on Mobile and Wireless Networking (MWN) 2003, held in conjunction with ICDCS 2003

Adaptive Power Control and Selective Radio Activation For Low-Power Infrastructure-Mode 802.11 LANs

Anmol Sheth, Richard Han
Department of Computer Science, University of Colorado
{sheth, rhan}@cs.colorado.edu

ABSTRACT

We present an integrated dual approach to reduce power consumption in infrastructure-mode 802.11 wireless LANs. A novel distributed power control algorithm adaptively adjusts the transmit power of the 802.11 radio/NIC to achieve power savings in the presence of mobility and RF noise. An adaptive radio activation algorithm selectively sleeps the 802.11 radio/NIC to reduce idle power consumption. We analyze the performance of these algorithms in an 802.11 testbed using Web traces of HTTP/TCP user data accesses. Our application-level architecture integrates the two approaches into an incrementally deployable and application-transparent solution for power management of the RF interface on 802.11-enabled laptops.

Keywords
Power control, 802.11b, WiFi, adaptation, power-aware, MAC, mobility, incremental deployment, sleep.

1 INTRODUCTION

Battery-powered wireless devices, such as cell phones, wireless PDAs, and 802.11-enabled laptops must conserve power. This paper focuses on developing a complete power management solution for 802.11/WiFi Network Interface Cards (NIC) configured in commonly used "infrastructure mode", i.e. for one-hop wireless access to the Internet. In infrastructure mode, a client laptop associates directly with a base station or access point and transmits/receives data packets directly to/from the 802.11 base station, which acts as a gateway to the Internet. 802.11 also offers an ad hoc mode for peer-to-peer communication between two laptops lacking a base station, but this mode is not considered in this paper.

The primary sources of power consumption on an 802.11 NIC are the duration of radio transmission when there are packets to send, the power level at which the radio transmits packets, and the amount of power consumed by the radio while it is idling waiting to receive packets. The duration of transmission can be reduced using techniques such as compression and aggregation. However, adjusting the transmission power and idling of the radio requires more complex interaction between the operating system and the 802.11 NICs. Idling has been shown to consume a significant amount of power in an 802.11 NIC [Feeney01, Shih02]. The objectives of this paper are three-fold: develop a practical algorithm to minimize the transmission power while retaining connectivity; develop a practical algorithm to minimize the power consumption of idling 802.11 NICs; and develop a system architecture that implements these two algorithms and provides a complete power management solution deployable to 802.11 users.

Determining the minimum transmission power that maintains connectivity is conceptually simple, but challenging to implement. Conceptually, the receiver simply subtracts the received signal strength indicator (RSSI) from the transmitted power of the packet to obtain the current path loss, and then informs the sender to transmit at (current path loss + P_{min}), where P_{min} is the minimum power at which the receiver can decode packets. In practice, the noisy RF reception environment results in fluctuations in the RSSI/path loss between packets even when both the client and base station are stationary [Sheth02]. The 802.11 client and/or base station could also be mobile, causing further fluctuations in RSSI. Therefore, the design of a practical bandwidth-efficient distributed algorithm for transmit power control must adapt to node mobility and RF noise/fading.

Minimizing the power consumption of an idle radio/NIC is conceptually simple, but challenging to implement in 802.11 LANs. The simple solution is to deactivate the radio when it is "not needed" for transmission or reception. In practice, it is non-trivial to determine when the radio should begin sleeping and how long the radio should sleep before waking up to receive packets from the base station. Packet arrivals are not deterministic, but instead vary according to user behavior and Internet delays. Since the client is not an oracle, then the client won't know precisely when to wake its radio to capture the new data packets just as they arrive from the base station. Infrastructure-mode 802.11 includes a power-save mode that allows the base station to queue packets for clients that have informed the base station that they will be going to sleep [IEEE99]. In this case, sleeping the radio would effectively delay packets, rather than lose them. Our challenge is to determine an algorithm that achieves a reasonable estimate of the next arrival for packets given variable delays and powers the radio up or down accordingly.

Our architecture for combining adaptive power control and selective radio activation should satisfy other design goals. We desire application transparency so that existing TCP/IP applications need not be rewritten to be power-aware, and can transparently benefit from adaptive power control and selective radio activation. We desire our solution to be incrementally deployable, so that 802.11 clients and base stations can continue to send data to one another if one or the other endpoint has not yet deployed our solution. Our design should have minimal impact on the 802.11 standard, requiring no changes to the packet structure, MAC protocol, and control signalling. Our system should broadly apply to 802.11, IR and other RF-modulated wireless systems.

Given these design objectives for power control, selective radio activation, and the system architecture, we describe the system architecture in Section 2, transmit power control in Section 3, selective radio activation in Section 4, and related work in Section 5.

2 SYSTEM ARCHITECURE

The first major design decision that we faced was determining at what layer in the protocol stack to implement transmit power control and radio deactivation. Our architectural solution consists of application-layer user-level processes that *observe* data traffic rather than modify data traffic, and then reliably communicate out-of-band power control updates via TCP/IP packets, as shown in Figure 1. A user-level process (Power-Aware PA module) monitors network traffic via a callback mechanism at each endpoint. The Berkeley Packet Filter packet capture library, which can be compiled into the kernel, inserts monitoring hooks and filters at various levels in the protocol stack. Our user-level process registers interest in data packets sent or received by the operating system via a filter, generating a callback to the user-level process containing a copy of the transmitted or received packet that satisfied the filter. These power-aware processes operate on a well-known TCP port to send and receive power control updates. Reliability ensures that the sender and receiver stay synchronized concerning the current agreed-upon transmit power level.

A key advantage of this user-level approach is that it is incrementally deployable. In case one or both endpoints do not speak our transmit power control protocol, then the power control updates are simply never sent, due to lack of a TCP peer process to receive the updates. Delivery of application data is not affected, except that data will be transmitted at non-optimal power levels. But if both a client and its infrastructure-mode base station speak our control protocol and support transmit power adjustment, then data will be exchanged at optimal power levels. Other architectural alternatives are not incrementally deployable. For example, several approaches covered in the related work section have proposed piggybacking their power control signalling on to the data packets, modifying each packet to contain its transmit power. Such an approach is not incrementally deployable, since the lack of a peer entity to interpret the modified packets would prevent data delivery; the receiver would either be looking for a field in the packet that didn't exist or would find a field in the packet that shouldn't be there, in either case causing the packet to be dropped; everyone would have to speak the protocol in order for data packets to be delivered.

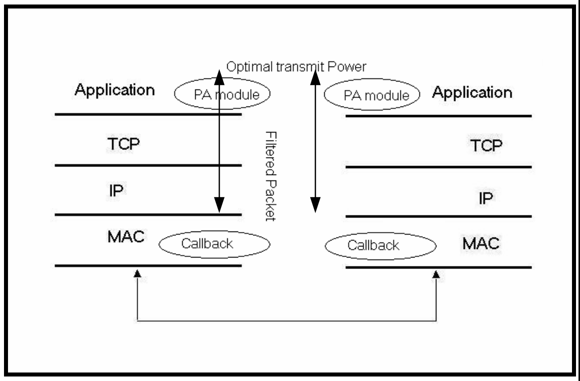

Figure 1. Our application layer architecture achieves application transparency and incremental deployment of power control.

Another advantage of our application-level approach is that it is inherently transparent to the application. The adjustment of the transmit power occurs outside of the flow of application data. TCP/IP applications such as Web browsers need not be rewritten in order to benefit from adaptive power control. Also, since the PA module can observe all TCP/IP packets to/from the 802.11 NIC, then the PA module is free to develop application-specific policies to adjust the transmit power and activate/deactivate the radio, e.g. Web-specific or email-specific policies for power control and radio activation.

Other benefits of our user-level approach include: minimal impact on 802.11 - no modifications to the existing 802.11 infrastructure, protocol, or packet structure are required, since our control signaling is user level and TCP/IP; easy porting of our application-layer TCP/IP solution to any wireless IR/RF system with adjustable transmit powers; easy upgrading simply by deploying new user-level PA modules to laptops and base stations.

A disadvantage of the user-level architecture is that PA modules are reactive in nature due to passive callback monitoring. The receiver's reactions are largely data-driven, and will lag the packet traffic's behavior. For instance, after a long silence, the first packet transmitted in a burst of packets will have a suboptimal power level.

We felt that this was a minor point given the other advantages of our architecture. Another potential limitation is that MAC layer signaling such as RTS/CTS may not be fully captured by user-level callbacks.

Our user-level architecture creates the same benefits for selective radio activation, i.e. applications such as Web browsing need not be rewritten to enjoy the benefits of selectively sleeping the radio/NIC. Selective radio activation works as a background process in concert with adaptive transmit power to achieve power conservation.

3 ADAPTIVE TRANSMIT POWER CONTROL

3.1 Algorithms

Our basic approach is to reduce the transmit power to the minimum level that still maintains connectivity despite intervening path loss, fading, and mobility. To determine this minimum level, the receiver computes for each packet the unidirectional path loss, equal to the difference between the transmitted power and the received signal strength (RSS). The optimal transmit power between a sender-receiver pair is given by:

$$\mathbf{P_{TxOpt} = Path\ Loss(t) + RSS_{min}} \quad \textbf{(1)}$$

where "Path Loss" includes multipath fading and shadowing. RSS_{min} is the minimum threshold that a packet can be correctly decoded by the radio (-80 dbm for Cisco Aironet 350 cards). In practice, the lower limit of the transmit power is 0 dBm for these cards, and P_{TxOpt} becomes the minimum of 1 mW and Equation 1.

The basic algorithm that we implemented for distributed calculation of P_{TxOpt} is shown in Figure 2. In our approach, the receiver B assumes the responsibility of calculating the path loss. If the sender A assumed responsibility for calculating path loss, then an additional packet would have to be transmitted back to the sender containing the RSSI, causing unnecessary power consumption on the receiver B. To achieve bandwidth efficiency, the receiver B only sends a control packet to the sender if an *event* has occurred, e.g. the RSSI has changed "significantly" due to mobility or there has been a timeout due to lack of data. Per-event adaptation utilizes bandwidth more efficiently than per-packet adaptation. In case of a timeout due to lack of data, our protocol conservatively suspects that mobility has caused a disruption and requests to increase the transmit power by 3 dBm for each timeout, up to the default maximum. We call this mobility adaptation mechanism "active pressure", since P_{TxOpt} is pressured upward by default.

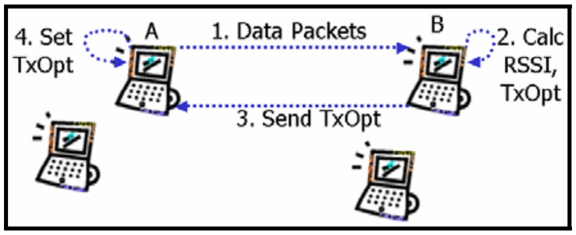

Figure 2: Basic algorithm for distributed calculation of the optimal transmit power. Example: laptop A sends data to base station B.

The algorithm is asymmetric or unidirectional, i.e. the calculation of T_{xopt}(A->B) differs from the calculation of T_{xopt}(B->A). This simplex approach has benefits when the path loss in both directions is not the same. In theory, the multipath reflections, shadowing and path loss should be symmetric, but in practice path losses we saw between two nodes in opposite directions were not symmetric, though there is some correlation.

Equation 1 is a tight bound and P_{TxOpt} would keep the sender and receiver just barely connected since packets would arrive near RSS_{min}. In practice, the RSS varies with time even at a fixed location due primarily to short-term multipath-induced noise. Figure 3 shows a trace of 802.11 RSS measurements collected while stationary (samples<30 and samples>100) and while there is mobility (30<samples<100). Since RSS variation causes path loss variation, then adhering to the tight bound of Equation 1 will result in unnecessary loss of packets whenever RSS momentarily falls below RSS_{min}. To avoid such packet losses, we placed a cushion M_{thresh} above the tight bound of Equation 1:

$$\mathbf{P_{TxOpt} = Path\ Loss(t) + RSS_{min} + M_{thresh}} \quad \textbf{(2)}$$

In [Sheth02], a fixed M_{thresh} cushion of 3 dBm above RSS_{min} was found to be large enough to prevent most below-threshold packet losses and was small enough to achieve considerable power savings through reduced transmit power. However, a fixed cushion adapts poorly when the RSS deviates widely around the average. M_{thresh} should track the deviation: when the deviation is small, then only a small cushion is needed above RSS_{min}; when the deviation is large, then P_{TxOpt} should have a large boost so that there is a large cushion above RSS_{min}.

To track the deviation, our algorithm computes an exponentially weighted moving average (EWMA) of both the RSS average, RSS_ave[n], and the absolute deviation in the RSS, RSS_dev[n], given an instantaneous estimate of the RSS for a received packet.

$$\mathbf{RSS\_ave[n] = \alpha{*}RSS\_ave[n-1] + (1-\alpha){*}RSS[n]} \quad \textbf{(3)}$$

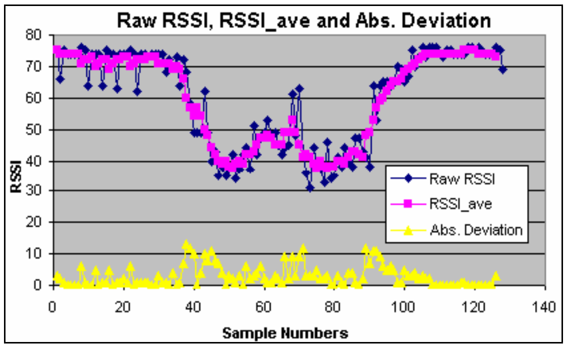

Figure 3. RSSI fluctuations are short-term (multipath) and long-term (mobility). Also shown: RSSI average and absolute deviation.

RSS_dev[n]=β*RSS_dev[n-1]+(1-β)*|RSS[n]-RSS_ave[n]| (4)

We replace M_{thresh} in Equation 2 with RSS_dev[n]. However, more exploration is needed to determine whether M_{thresh} should be a multiple of RSS_dev, e.g. 4*RSS_dev as in TCP [Stevens94]. Examples of the smoothed RSS_dev and RSS_ave are shown in Figure 3, where RSS_ave was calculated using a value of α=0.7, which biases the estimate towards the historical mean rather than the most recent value.

While Equations 2-4 provide a framework for adapting to short-term multipath-induced fluctuations, they also provide a framework for adapting to "significant" mobility-induced events. Figure 3 shows that mobility causes long-term changes in the RSS average. Our algorithm detects these "significant" changes in the RSS average by comparing the current RSS_ave with the "pegged" value of RSS_ave from the last time a power control update was sent to the transmitter. If the difference is significant, then a new control update is sent to the transmitter to adjust its transmit power, and we save RSS_ave, RSS_dev, and the *j*'th adjustment to the optimum transmit power P_{TxOpt}[j]. Pseudo code resembles the following: if (RSS_ave[n] - RSS_ave[last_pegged]) > threshold, then {last_pegged = n, save RSS_ave[last_pegged], recalculate new P_{TxOpt}[j], and send it to transmitter}. The threshold chosen to trigger the "significant" long-term change remains fixed at 2 dBm, which corresponds to the falloff in signal strength over about 10 m, which is a distance a walking individual could reasonably traverse at a rate of 1.5 m/s [Sheth02]. The precise formula for P_{TxOpt}[j] at time *n* given *j-1* previous adjustments of the transmit power is:

P_{TxOpt}[j] = P_{TxOpt}[j-1] - RSS_ave[n] + RSS_{min} + RSS_dev[n] (5)

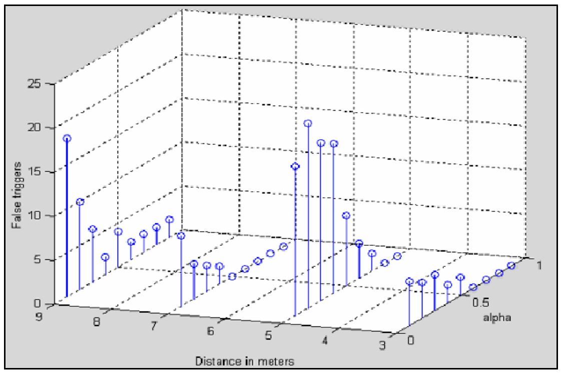

Figure 4. False triggers for alpha varying from 0.1 to 0.9 and source transmitting at 13 dBm

This framework minimizes the transmit power while also adapting to short-term multipath and long-term mobility, and does so in a per-event bandwidth-efficient manner.

3.2 Experiments and Performance Analysis

We constructed experiments to determine optimal values of α and β. Since α affects RSS_ave, and differences in RSS_ave are used to trigger mobility, then we wish to choose an α that minimizes the number of false triggers when short-term multipath-induced fluctuations in RSS are mistaken for long-term fluctuations due to mobility. We collected traces of RSS samples at stationary distances of 3m,5m,7m and 9m between a source and a destination node, i.e. separate 802.11 laptops. The transmit power was fixed. The base station's firmware was inaccessible, so measurements were taken in ad hoc mode, which should not affect RSS.

For each distance and for each value of α ranging from 0.1 to 0.9 in steps of 0.1, we counted the number of false positives, i.e. the number of times when mobility is detected (RSS_ave[n]-RSS_ave[last_pegged]) > 2 dBm), though there is no true motion. Figure 4 reveals a clear pattern across all distances in which the number of false triggers decreases with increasing α. Values of α from 0.7 to 0.9 resulted in the fewest false positives. This is expected, since higher values of α insulate the EWMA mean from temporary fluctuations that could be mistaken for mobility. A high value of α (0.9) would cause RSS_ave to respond sluggishly to rapid changes attributable to true mobility. Hence we chose an optimal value of α = 0.7, which remained consistent across all transmit powers (20dBm/100mW, 17dBm/50mW and 13dBm/20mW)

To determine the optimal β value, we observe that β affects RSS_dev, which is used to define the cushion above RSS_{min}. A poor choice of β would not provide enough cushion, resulting in lost packets when RSS dips below RSS_{min}. We set an artificial RSS_{min} to 0dBm (lowest transmit power that can be set on the card) and counted the number of times the RSS dips 0dBm for different values of β. We observed that lost packets were lowest for β >= 0.7. We selected an optimal β = 0.7.

Figure 5 demonstrates the motion detection points generated by the adaptive algorithm using a value of α = 0.7. The trace is identical to Figure 3, e.g. static then mobile then static, but only the RSS_ave curve is plotted. The trigger points at which motion is detected are shown as black squares and agree closely with intuition.

To measure the power savings of this power control algorithm, we sampled the voltage drop across the wireless card using a DAC, and observed that a *maximum* energy savings of 25% can be achieved, though these results are affected by idle energy consumption. If transmission power is isolated from idling power, then the potential for savings is dramatic. On most occasions, the card's transmit power could be reduced from a default value of 20 dBm (100 mW) down to 1-10 mW for distances less than 30m, a savings of 90+ % during transmission [Sheth02].

4 SELECTIVE RADIO ACTIVATION

In this section we present our adaptive protocol for selective activation of the radio to reduce idle power consumption when a host is dormant. The infrastructure mode of the 802.11b MAC protocol specification supports a power save mode that allows a host to specify to the base station how many time intervals that the host wishes to sleep. A host will specify the total sleep interval, also known as the Listen Interval, as an integral multiple of the Beacon Interval, which is the wait time between periodic broadcasts of the base station's beacon. The base station buffers packets arriving during the Listen Interval for the sleeping host. Our algorithm exploits this feature to adaptively specify longer sleep intervals when there is light traffic and shorter sleep intervals when there is heavy traffic.

Ideally, our algorithm should sleep the radio only when there are no packets to be transmitted or received. Determining when to wake the radio is simple for transmitted packets - as soon as a packet needs to be sent, then the radio is awakened. Determining when to wake the radio in anticipation of receiving packets from the base station is more difficult. We confine our studies to actual Web traces. Figure 6 shows that Web behavior is bursty. A large number of packets are sent within a burst over a relatively short time, i.e. a Web page is downloaded. In between bursts, there are long interarrival times between packets, i.e. the user is reading a Web page. Our key contribution is to determine *when* a radio should go back to sleep after servicing a burst of packets, i.e. determining the *end* of the burst.

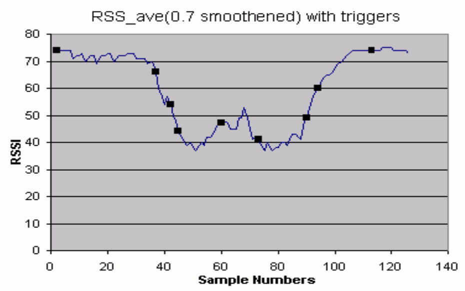

Figure 5. RSS_ave with trigger positions detecting true mobility.

To determine the end of a packet burst for Web traffic, we construct per-server EWMA interarrival times (IAT). We assume that the Web browser is on the host laptop pulling data from a remote Web server through the base station. Intuitively, after a "significant" silence, the host times out, believes that the burst has ended and goes to sleep. We calculate the timeout from EWMA smoothed estimates of the average IAT (IAT_ave) and absolute deviation of the IAT (IAT_dev), reasoning that instantaneous measurements of IAT are too noisy. If the timeouts are based only on the most recently observed IAT, the timeout may be too short, leading to the NIC waking up prematurely and consuming idle power until packets arrive at the base station. Timeouts that are too long result in packets being queued at the base station, causing a delay. The equations for calculating the IAT mean and the variance are the same as (3) and (4). The timeout should track the average - if no packet has arrived by the average IAT, then the burst from this server has likely ended. However, if there is significant variability in IATs, then we should wait a little longer to account for those packets with longer IATs. We use the following formula to compute the per-server timeout:

$$\mathbf{t_o = IAT\_ave + IAT\_dev} \qquad \mathbf{(6)}$$

Since the components of a given Web page may arrive from many Web servers, we maintain timeouts t_o, IAT_ave, and IAT_dev for each server and select the maximum t_o as our overall timeout, i.e. we wait for the slowest server to respond with its data before timing out and going to sleep. Since some wireless NIC's, have a very high startup time T_s, we integrate this into our timeout, as explained next. The final timeout T_o equals:

$$\mathbf{T_o = max\{ maximum\ of\ all\ t_o,\ T_s\}} \qquad \mathbf{(7)}$$

When T_o expires, IAT_ave and IAD_dev are preserved or pegged, and are not updated between bursts.

After timing out with T_o, we need to determine how long to sleep before waking up for the next burst. We investigated two techniques, additive increase and

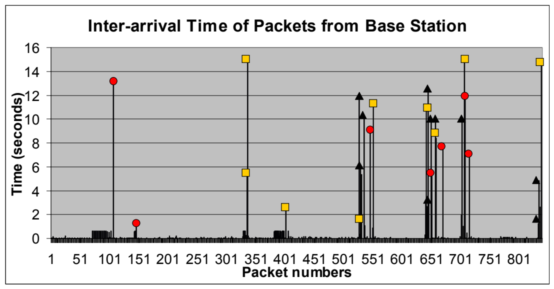

Figure 6. Inter-arrival time of packets arriving from a base station (Web trace).

● SYN + ACK, 🟨 ACK for FIN, ▲ FIN

exponential increase, that adaptively probe to see if there are any queued packets at the base station and if not increase the sleep time. Additive increase produces sleep durations like T_o, $2T_o$ $3T_o$, $4T_o$.... Exponential increase doubles the sleep duration: T_o, $2T_o$ $4T_o$, $8T_o$.... Suppose the maximum t_o is 100 ms while T_s is 2 sec (Aironet). In exponential increase, we'd sleep "virtually" for the first T_o = 2 seconds, i.e. the card is never powered down, then sleep $2T_o$ (2 sec real sleep, 2 sec startup), then sleep $4T_o$ (6 sec real sleep, 2 sec startup), etc. If instead, the maximum t_o is 100 ms while T_s is 5 ms (Prism 2), then we'd timeout after (or sleep "virtually" for) 100 ms, then sleep for 200 ms (195 ms sleep, 5 ms startup), etc.

Delay was calculated by measuring the amount of time the node remains in low power sleep mode after packets have arrived at the base station. The energy consumption was calculated by summing the energy for waking up, querying the base station and sleeping in low power sleep mode. The experiments used 5 minute snapshots of user Web traces, and a wireless NIC with PRISM 2 chipset. The deepest sleep mode (level 4) consumes 30 mA, takes a negligible 5ms to switch to active mode and consumes on an average 150mA. In the idle mode the card consumes 290mA. We found that packets in additive increase experienced a smaller average queuing delay (7.5 seconds) than exponential increase (34.66 sec), since additive increase wakes more often to check for queued packets. However, additive increase consumed more energy (53.88 J) than exponential increase (33.22 J), again because additive increase wakes more often. Our key observation, as shown in Figure 6, is that the kinds of packets delayed are most often FINs and FIN-ACKs that terminate a TCP connection. Such packets can suffer significant delay without affecting the user's interactivity. The less common case was that some packets are SYN-ACKs, i.e. delayed responses to requests to open a Web page. For SYN-ACK packets, the delays will be noticeable if the radio is sleeping. We select exponential increase over additive increase because of its improved energy savings and because the increased delay does not materially affect the most common form of delayed packets, namely FINs and FIN-ACKs, and accept the delay for SYN-ACKs.

We calculated the timeouts from the smoothed IATs within a burst for different α and β to find the combination (α, β) that minimized the queuing delay. We extracted short bursts (1-1.5 sec) to each Web server. The number of packets delayed is minimum for α = 0.9 and β = 0.1 consistently across servers.

5 RELATED WORK

Power control is employed to increase network traffic capacity [Elbat02, Monks01], vary levels of service and reliability [Kandu01], and reduce power consumption [Agarw01]. Monks et al. proposed a distributed power control algorithm for 802.11 wireless LANs to improve the network's capacity, i.e. MAC layer collisions are reduced by reducing the transmit power to the minimum level to maintain connectivity [Monks00, Monks01]. This work was confined to ns2 simulation, doesn't address mobility or RF noise, and assumes separate channels for data and control, thereby requiring changes to the 802.11 standard. The COMPOW protocol [Naray02] maintains that network capacity is asymptotically maximized by selecting a common minimum transmit power for all nodes equal to the minimum power at which the network maintains connectivity. This approach is implemented using Cisco Aironet 350 NIC's, but requires modification to MAC packet headers and is not adaptive to RF noise. Agarwal et al simulate distributed 802.11 power control that includes a mobility model [Agarw01]. This scheme computes an EWMA of the RSSI, but depends upon modifications to both the 802.11 CTS control packets and DATA packets. Also, our EWMA scheme: 1) calculates the optimal transmit power immediately for each packet, rather than stepping up or down towards the optimum power level until a packet is lost; 2) updates the transmit power when there has been a significant change, rather than waiting for ten packets after a loss, and thus immediately adapts after fading to an improved channel.

Jung et al. propose transmitting the RTS/CTS at maximum power while also intermittently transmitting data packets at maximum power, in order to prevent collisions [Jung02]. The approach does not address mobility and also makes the assumption that path loss is symmetric between two nodes. Topology control/management and cross-layer optimization address how heterogeneous transmit powers affect the connectivity of a multi-hop wireless ad hoc routing network. [Elbat00, Gomez01, Kwon99, Ram00]. We focus on single-hop infrastructure-mode operation.

To address the problem of idling power consumption in 802.11 NIC's [Feeney01], the client's radio is selectively powered down based on user/application inactivity [Krash02, Kravets00, Simun00], transmissions from wireless neighbors [Singh98], or a wake up message from a base station via a 2nd radio [Shih02]. Kravets et al deactivate the radio after a fixed application-dependent timeout and suspend the radio for an application-dependent sleep time. The radio wakes and if no packets are queued goes back to sleep at double the previous sleep time, up to a 5 minute maximum. This improves power savings and reduces queuing delay compared to a fixed sleep duration for Web traces. Our approach computes an EWMA timeout rather than a fixed timeout and maintains application transparency. Simunic et al employ a Time-Indexed Semi-Markov Decision Process (TISMDP) to predict user Web request arrivals and then decide whether to power down the 802.11 card [Simun00]. TISMDP conserves power close to the performance of a perfect oracle for a given Web trace. The paper is unclear whether TISMDP predicts only the arrival of HTTP GET requests for transmission, or also predicts the interarrival times of HTTP responses received by the radio. Since current radios can take up to two seconds to completely wake, then anticipating the arrivals of HTTP requests and responses is crucial to give the NIC enough time to wake up before it is needed. Krashinsky et al propose a Bounded Slowdown (BSD) protocol that adaptively suspends an 802.11 radio in power-save infrastructure mode, but only if no roundtrip time (RTT) is lengthened by more than a tunable factor p [Krash02]. BSD consumes less energy than a periodic power-save algorithm and incurs only slightly more delay than an always-on radio when accessing Web data. The algorithm at most doubles the roundtrip time between wakeup times. However, BSD uses a fixed not an adaptive timeout, and does not account for the finite startup time of the radio.

6 CONCLUSION

This paper presents a complete power management solution for 802.11 infrastructure-mode LANs. Our application-layer system architecture achieves application transparency, incremental deployment, and minimal impact on 802.11. Our power-conserving bandwidth-efficient algorithm for transmit power control adapts to RF noise and mobility. Power conservation via selective activation of the radio times out after an exponentially-smoothed timeout and sleeps between exponentially increasing probes. Both techniques were evaluated via Web traces.

7 REFERENCES

[**Agarw01**] S.Agarwal, S. Krishnamurthy, R. Katz, and S. Dao, "Distributed Power Control in Ad Hoc Wireless Networks", IEEE Personal, Indoor and Mobile Radio Communications (PIMRC) 2001, vol. 2, pp. F-59 -F-66.

[**Elbat00**] T.Elbatt, S.Krishnamurthy, D.Connors and S.Dao, "Power Management for Throughput Enhancement in Wireless Ad Hoc Networks," International Conference on Communications (ICC) 2000, vol. 3, pp. 1506-1513.

[**Elbat02**] T.Elbatt and A.Ephremides, "Joint Scheduling and Power Control for Wireless Ad Hoc Networks," INFOCOM 2002, vol.2 , pp. 976 -984.

[**Feeney01**] L. Feeney, M. Nilsson "Investigating the energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment" IEEE INFOCOM 2001, vol. 3, pp. 1548-1557.

[**Gomez01**] J. Gomez, A. T. Campbell, M. Naghshineh and C. Bisdikian, "Conserving Transmission Power in Wireless ad hoc Networks", IEEE 9th International Conference on Network Protocols (ICNP'01), 2001, pp. 24 -34.

[**IEEE99**] IEEE802.11b/D3.0. Wireless LAN Medium Access Control (MAC) and Physical (PHY) Layer Specification: High Speed Physical Layer Extensions in the 2.4 GHz Band. 1999.

[**Jung02**] E. Jung, N. Vaidya, "A Power Control MAC Protocol for Ad Hoc networks", Mobicom 2002.

[**Kandu01**] S. Kandukuri, N. Bambos "Multimodal Dynamic Multiple Access (MDMA) in Power Controlled Wireless Packet Networks." INFOCOM 2001, vol. 1, pp. 199-208.

[**Krash02**] R. Krashinsky, H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown", MobiCom 2002, pp. 119-130.

[**Kravets00**] R. Kravets, P. Krishnan, "Application-driven Power Management For Mobile Communication," Wireless Networks, no. 6, 2000, pp. 263-277.

[**Kwon99**] T. Kwon and M.Gerla, "Clustering with Power Control.," IEEE Milcom, 1999, vol. 2, pp. 1424 -1428.

[**Monks00**] J. P. Monks, V. Bharghavan, and W. Hwu, "Transmission Power Controlled for Multiple Access Wireless Packet Networks," IEEE Conference on Local Computer Networks (LCN), 2000, pp 12-21.

[**Monks01**] J. P. Monks, V. Bharghavan, and Wen-mei Hwu, "A Power Controlled Multiple Access Protocol for Wireless Packet Networks," IEEE INFOCOM 2001, pp 219-228

[**Naray02**] S. Narayanaswamy, V. Kawadia, R. Sreenivas, and P. Kumar, "Power Control in Ad Hoc Networks: Theory Architecture, Algorithm and Implementation of the COMPOW protocol", European Wireless: Next Gen. Wireless Networks..., 2002, pp. 156-162.

[**Ram00**] R. Ramanathan and R. Rosales-Hain "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment" IEEE INFOCOM, 2000, vol. 2, pp. 404-413.

[**Sheth02**] A. Sheth, R.Han, "An Implementation of Transmit Power Control in 802.11b Wireless Networks", University of Colorado, Department of Computer Science, Technical Report, CU-CS-934-02, August 2002.

[**Shih02**] E. Shih, P. Bahl, M. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," MobiCom 2002, pp. 53-64.

[**Simun00**] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Dynamic Power Management for Portable Systems", MobiCom 2000, pp: 11-19.

[**Singh98**] S. Singh and C. Raghavendra, "PAMAS: Power aware multi-access protocol with signalling for ad hoc networks," ACM Computer Communication Review, vol. 28, no. 3, pp. 5–26, July 1998.

[**Stevens94**] W. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, 1994.