Exploiting Physical Layer Power Control Mechanisms in IEEE 802.11b Network Interfaces

Marco Gruteser, Ashish Jain, Jing Deng, Feng Zhao, and Dirk Grunwald

CU-CS-924-01 December 2001

University of Colorado at Boulder

Technical Report CU-CS-924-01
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309

Abstract

Mobile computing relies on low-power communication. Since ad-hoc networks are often idle, many researchers have investigated mechanisms to reduce idle power. This paper presents power measurements on current IEEE 802.11b network cards and discusses physical and link layer power control mechanisms for *transmission power*. Specifically, we characterize the influence of bit-rate, transmission power, and packet size on total power consumption. These measurements indicate a high per-packet overhead, indicating that packet aggregation might be able to reduce that overhead. Packet aggregation combines multiple smaller packets into a larger packet. Our experiments show that our application-independent packet aggregation mechanism yields substantial transmission power savings for workloads that frequently send small packets, such as Web workloads.

1 Introduction

Energy has become a central design constraint in mobile computing. Especially, as technology progresses to ever-smaller computing devices, the significance of energy consumption in the wireless network interface increases. For example, the power consumption of an IEEE 802.11 network interface can make up 50% of a handheld computing device [1]. Therefore, wireless network energy consumption has been the focus of many low-power computing research projects.

Many applications use the network interface only sporadically. Since, idle power consumption on network interfaces is high, most previous work has focused on card deactivation during idle phases [2] [1]. With improvements in wireless network card design, idle power consumption is becoming less of a problem. For example, the Cisco Aironet 350 series card [3] has a specified sleep power consumption of only 50mW. In addition, the idea of a "paging" mechanism has the potential of virtually eliminating this idle power consumption.

In this paper, we focus on the problem of *active transmission power consumption*, including both transmission and reception. We develop a model of the transmission overhead for varying packet sizes and used this to implement an application-independent packet-aggregation protocol that reduces transmission and reception overheads while maintaining quality of service constraints. The key contributions of this paper are:

- Characterizing the influence of bitrate and transmission power on network interface power consumption
- Describing the per-packet overhead in modern 802.11b network interface
- Demonstrating the feasibility of application-independent packet aggregation

Packet aggregation combines multiple small packets into one big packet before transmission over the wireless link. We will show that this results in transmission energy savings of up to 75%, while imposing a mean delay of 100ms onto packets.

The rest of this paper is structured as follows: First, we will discuss related work in power-aware networking. We describe the important aspects of the IEEE 802.11b standard in 2.1. In section 3 we describe our experimental methodology for measuring network card energy consumption. We measured how bitrate and transmission power affect power consumption and how packet sizes influence transmission energy and power. We then see how packet aggrgeation can exploit these attributes to reduce active power.

2 Related Work

Energy efficiency is not only an issue for different components in mobile computers such as network, CPU [4], and hard disks [5], it is also addressed at different levels. Network power management work can be categorized into media access control mechanisms and operating system mechanisms from a single-node and a network perspective.

The media access control (MAC) sublayer generally has a role of reducing collisions and scheduling access. This information can influence the ability of the radio to control energy and QoS. For instance, as shown in [6, 7], the MAC layer can be aware of nearby radios transmissions schedules when the channel will be busy and so can simply shut down during these intervals in order to save power. Cellular systems also use segregated control and data channels to minimize contention inefficiencies [8, 9]. Similarly, Singh [6] proposes a separate contention and data channel.

Transmission power control has first been investigated from the perspective of network capacity [10] [11]. Transmission with reduced power cause less interference for other nodes in the network. Thus, total network capacity increases. From an energy consumption perspective this issue is presented in [12].

Kravets and Krishnan [13] as well as Stemm and Katz [2] have used operating-system controlled admission scheduling to reduce idle power consumption. While Stemm's results where simulation based, Kravets' implementation used relatively simple beacon scheduling to coordinate system activity and the performance of applications was reduced significantly. The system was divided into a master/slave organization with a centralized control point, and no discussion was given to distributed control. Furthermore, good performance depended on the operating system knowing good control parameters for each application. Stemm also measures the effect of different transport protocols on a Metricom Ricochet Wireless Modem and a pre-802.11 AT&T Wavelan card. Detailed measurements for network operations on 802.11 and 802.11b Wavelan cards are presented by Feeney et al. [14].

Recently RL has been demonstrated on link layer adaptation in wireless networks [15]. The controller was able to learn a policy that meets QoS targets while also reducing the unnecessary energy use by 50% to 80%. The radio controller monitors packets sent and received over the wireless interface and learns a policy for turning the radio on and off.

Other studies have used transcoding or addressed systems that admit loss-allowing communication protocols [16, 17, 18] by adapting the protocol to changes in the data link layer.

The goal of the network layer is to maintain connectivity to all participants for the longest period while meeting QoS guarantees. In [19], Singh analyzes link metrics that lead to increasing the life of nodes. Appropriate metrics lead to a factor of two reduction in energy use for the worst case user. This study considered a random graph and not one based on physical locations. More realistic ad hoc network studies include terminals distributed across the plane. Takagi [20] considers the optimal transmission range to maximize packet progress. Larger powers mean more connectivity, but more terminals for collision and vice versa. If transmitter power can vary from node to node, connection to connection, then the minimum power to remain connected can be computed [21]. In [22], the authors consider the general problem of disseminating information for limited energy ad hoc networks of sensors. They show that the routing protocol can be optimized for this problem to yield lower energy consumption. Varying transmit power, the link metrics used for the resulting connectivity, and optimized protocols can be combined to simultaneously guarantee connectivity and extend the battery life of the worst case participant.

Additionally, in [23] the concept of data fusion is proposed. Data from multiple nodes is combined before it is forwarded to the next routing hop. Our packet aggregation approach is more generally applicable, because aggregation nodes can operate at the network packet level. They are not required to know the protocol and data semantics used in these transmissions.

The problem of idle power consumption from the network perspective has been addressed by Chen et al. [24]. Deactivation of network interfaces are coordinated within the network, so that networks are not partitioned and bandwidth and delay requirements are observed.

2.1 Background: IEEE 802.11b

The IEEE 802.11b standard provides a standard for the ISO physical and link layers of high-throughput wireless networks. It supports variable transmission power and bitrates of 1, 2, 5.5, and 11 Mbps. Two operating modes are defined: infrastructure and ad hoc mode. In infrastructure mode, all radios associate with a base station, whereas in ad hoc mode they communicate directly. In the physical layer, spread-spectrum radio techniques are used within the 2.4 GHz ISM band. This band is reserved for unlicensed operation in most countries. Spread-spectrum techniques reduce interference with uncooperative devices. The 11Mbps throughput was achieved through abandonment of frequency hopping and improvements in data encoding. Quadrature Phase Shift Keying uses the radio bandwidth more efficiently and Complementary Code Keying improves data coding.

The link layer is based on the IEEE standards for wired networks. However, the wireless environment made modifications necessary. Since radio stations cannot send and transmit at the same time, the CSMA/CA collision

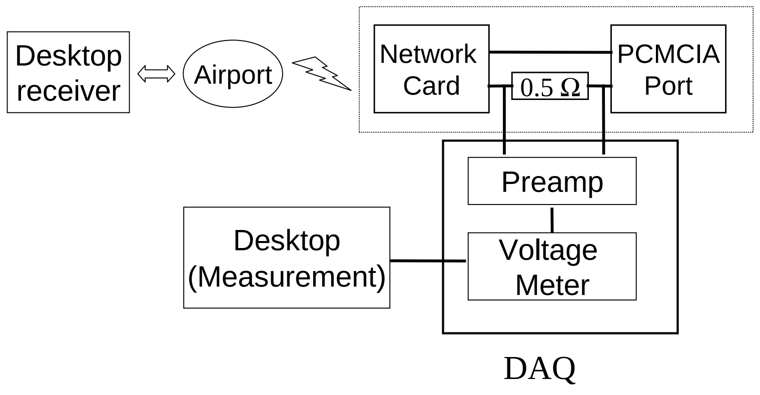

Figure 1: Measurement setup

Resistor (mOhm)	100	500	1000
Idle Power (W)	1.2636	1.2732	1.2119

Table 1: Variance with different resistors

avoidance protocol is used instead of CSMA/CD. The 802.11 specification also supports a "request to send/clear to send" mechanism to avoid the hidden terminal problem, but this mechanism is typically disabled in most implementations[1]. To compensate for higher error rates, checksums and explicit acknowledgements were introduced into the MAC layer, which cause significant overhead compared to wired networks.

3 Equipments & Methodology

In this section, we describe our general measurement setup as illustrated in figure 1. Details for specific experiments are given in the experiment sections.

The energy consumed at the network card is the integral in time over power. Power is the product of voltage and current. The voltage at the PCMCIA card is known to be either a constant 5V or 3.3V. However, we need to measure the changing current. Since our measurement station only takes voltage samples, we introduce a known resistor of 0.5 ohm in series with network card and measure the voltage drop across this resistor. We can then calculate the current using Ohm's law, resulting in $E = \int \frac{(V_{cc} - V(t))V(t)}{R} dt$.

We measure network card power consumption on the PCMCIA port of a *Dell Inspiron 3800* laptop. A *Kaitek Engineering Cardbus Extender* card is inserted between the PCMCIA port and the network card to give us access to the power supply of the card. Our measurement equipment provides a varying sample rate; we found that a 20Khz sampling rate eliminated most measurement errors. The measurement station is comprised of a *National Instruments DAQ BNC-2090* and a Linux machine running the *Comedi* interface v0.7.60 [25] and *Ktimetrace* [26] utility. The *Ktimetrace* utility creates a log of the voltage samples in a simple ASCII file format. Energy, average current, average power, and peak power consumption is calculated from this log with a *Perl* script.

To ensure that we chose a suitable resistor, we compared the variance from measurements with different resistors. If the resistor is chosen too small, the voltage drop cannot be accurately measured with our equipment. However, if it is too large, it might affect the power consumption or even the operation of the network card. We experimented with resistors in the range of 0.1 to 1 ohm. The results, averaged from three 10s measurements are presented in table 1. The variance is less than 5%.

To completely verify our measurement setup we compared measurements of an Aironet 350 series card with manufacturer specifications [3], which are presented in table 2. Our measurements deviate less than 5%.

[1]In particular, the RTS/CTS mechanism is only used for packets above a specific threshold, and the default threshold is large than the default MTU for 802 networks

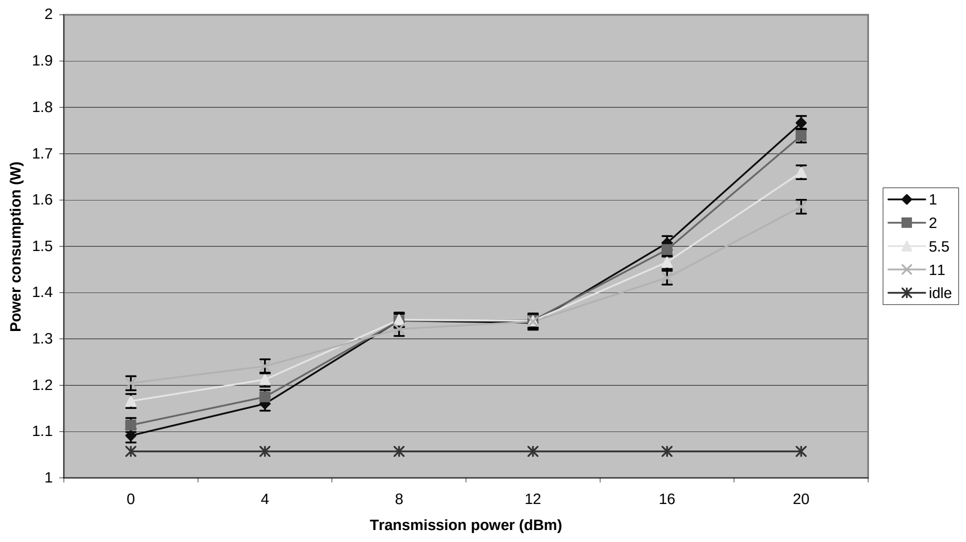

Figure 2: Influence of bitrate and transmission power on power consumption

4 Effect of Bitrate and Transmission Power

We configured a network card into every possible combination of transmit power and bitrate and recorded power measurements while it was actively sending. We used *Netperf*[27], a standard TCP benchmarking tool, to keep the card transmitting during our measurement. The average power is obtained from 10s measurements. Every experiment was repeated three times and we calculated 95% confidence intervals from this data. Although this is very few samples and would normally result in very large error bars, the variance was exceptionally small between samples and there was little reason to collect more samples.

We used a Cisco 350 series PCMCIA 802.11b for this experiment. To our knowledge, the Cisco 350 series has the most fine-grained control of transmit power. It has six different transmit power levels: 1mW, 5mW, 20mW, 30mW, 50mW, and 100mW. It also supports the standard IEEE 802.11b bitrates: 1Mbps, 2Mbps, 5.5Mbps, and 11Mbps.

We were unable to force the access point and 802.11b cards to use a specific bit rate in infrastructure mode; when the cards were forced to used a specific bit rate, they lost connectivity with the access point (an Apple Airport V1.0). Thus, for this measurement only, we conducted the experiments in *ad hoc* mode, which allowed us to vary the bitrates using the device drivers. We conducted the experiment in ad hoc mode with a Linux driver. We experimented with the Microsoft Windows98 driver because of better manufacturer support; however, we were unable to select different bit rates using this driver.

We placed two laptops with Cisco 350 series cards approximately 10 feet apart. We set up the measurement hardware and a *Netperf* client on one machine and a *Netperf* server on the other laptop. In the following description, we refer to those machines as client laptop and server laptop, respectively. We configured the bit-rate and transmit power on the client laptop with the *iwconfig* utility from *Linux wireless extensions*.

The results of our measurements are shown in figure 2. The straight line at the bottom shows the idle baseline. Each of the other lines represents the measurements for one bitrate setting. As expected, the lowest power consumptions were encountered at 1mW (=0dBm) transmission power. At this level, they are very close to the idle baseline. From there, they gradually increase by about 50% to the highest power consumptions at 100mW (=20dBm).

Mode	Transmit	Receive	Sleep
Power (mW)	1750	1250	50

Table 2: Cisco Aironet 350 series manufacturer specifications

The effect of bitrate on power consumption is more complex. Surprisingly, at the high transmit power levels faster bitrates have lower average power consumption then slower bitrates. The lines cross at medium transmit power levels and at lower levels, the faster bitrates have higher power consumption. We conjecture that the lower bitrates consume more power than higher bitrates at high transmit power because of differences in bitrate utilization. While *Netperf* maximize transport-level throughput for all measurements, the throughput data show that there are differences in link-level utilization. It is much closer to the theoretic maximum throughput at lower bitrates. Therefore, the card should be actively sending for a higher percentage of time at lower bitrates compared to higher bitrates. If the card is not sending, then the power consumption will be at the idle level. Ergo, at higher bitrate the card spends more time idling and thus the average power consumption is lower. Why does this not hold for lower transmission power levels? We speculate that this is due to different amounts of packet loss. As we lower the transmit power, packet loss probably becomes more significant. In addition, we expect the amount of packet loss to be dependent on the bitrate. Higher packet loss at higher bitrates would cause more resends and therefore higher power consumption.

These results indicate that significant overall *power* savings can be achieved by lowering the transmission power and that varying the bit rate has less impact on the overall power savings.

Whether these power savings could be translated into *energy* savings is unclear; if an application must transfer a specific amount of data, our data indicates that is is more useful to transmit the data at the highest bitrate possible, particularly if the idle mode power can be reduced. If the application transfers data at a fixed *rate* (such as a media application), it may be that reducing the bitrate can reduce the overall energy, particularly if the transmission rate prevents the effective use of idle energy reduction. Likewise, the transmission power level must be adjusted to reduce the bit error rate least packet retransmissions eliminate any energy savings.

Reducing the power level to the minimum required to reach the receiver also causes less interference. Since the signal is weaker, nodes that are further away can operate undisturbed. To our knowledge, the Cisco 350 cards are already capable of automatically adjusting transmission power. However, the results suggest another power management strategy. The effective range of the wireless signal can be increased by either increasing the transmit power or lowering the bitrate [3]. The data show that the latter is more power-efficient. Hence, adaptively varying the bitrate based on the signal strength and the bandwidth required by applications may result in additional power savings.

5 Effect of Packet Size

We explored the effect of packet size on the power overhead. For older network cards; this relationship has been explored by Feeney [14]. We verify that the results still hold on two more recent cards with the following experiment. We also generate a transmission overhead energy model for our following experiment.

We sent a fixed number of same-sized packets during a 30s interval and measure the total energy consumed. One packet is sent every 40ms. This experiment is repeated with packet sizes of 50, 100, 250, 500, 1000, and 1450 bytes. To compare against the baseline power consumption, we also take measurements while the card is idle. We implemented a packet generator in the Click Modular Router Framework [28]. The Click Framework simplifies the construction of routers through a modular approach. We used Click because it simplifies the implementation of packet manipulation code and we later used the same infrastructure for packet aggregation. Our packet generator sends Ethernet frames of specified size and at a specified rate. We conducted this experiment on the Cisco 350 with the linux driver and an SMC 2632W using the *linux-wlan-ng* driver for the Intersil Prisim chipset. Again, we repeat the measurement for every configuration three times and show 95% confidence intervals.

Figure 3 shows the average power measured for different packet sizes. We also calculated the energy per byte dependent on packet size. We arrived at these numbers by first subtracting the idle energy from our measurements. Therefore, the results shown in figure 4 only represent the actual *transmission overheads*.

Comparing the results in figure 3 , we see that the Cisco 350 card consumes about 10% less idle energy than the SMC 2632W. Furthermore, the per-byte transmission overhead (*i.e.*, the additional power needed to transmit each additional byte as the packet size is increased) of the Cisco card is an order of magnitude lower than that for the SMC card.

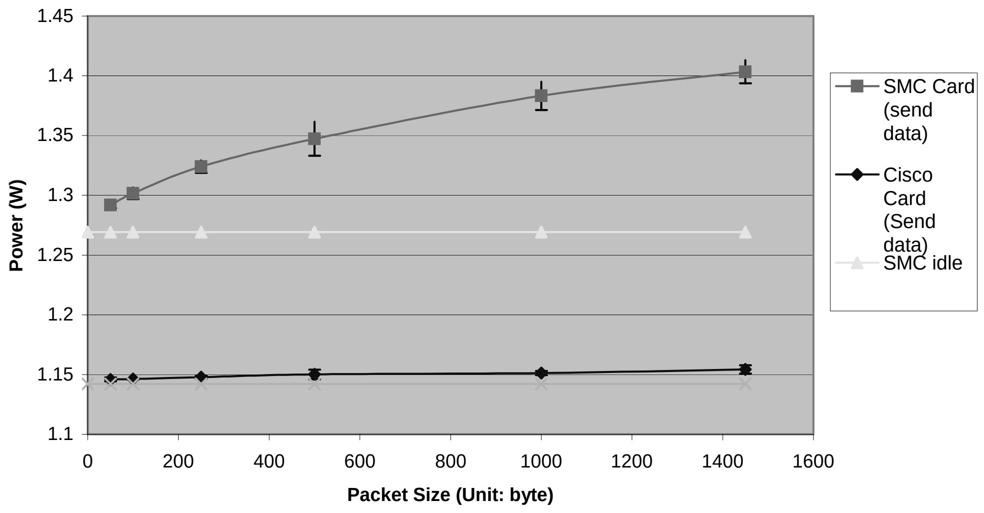

Figure 3: Influence of packet size on power consumption

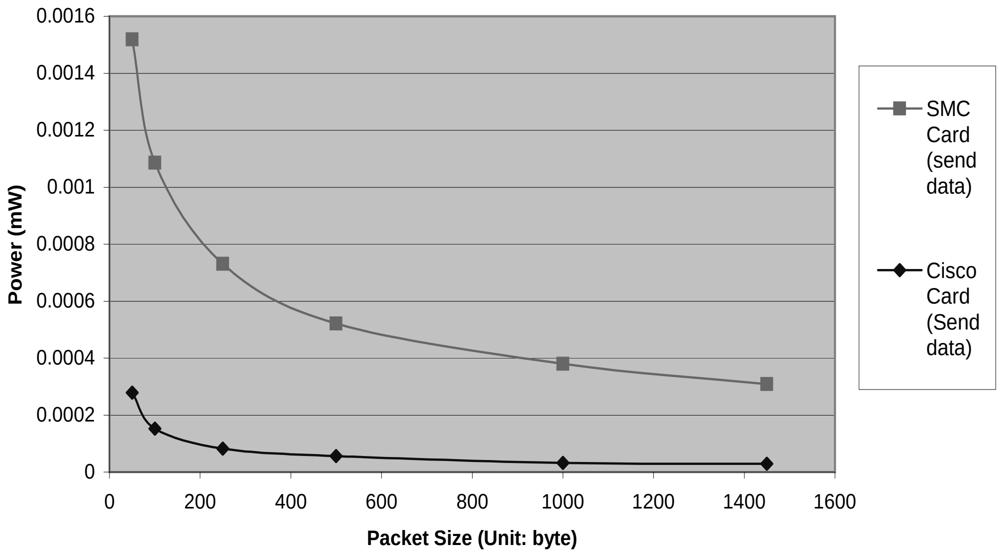

Figure 4: Energy/byte at different packet sizes

As expected, transmission overhead increases for larger packets. However, even small packets exhibit a significant overhead. From figure 4 we conclude that the transmission energy overhead per byte decreases significantly for larger packets.

For the Cisco 350 card, the packet transmission energy (in Joules) can be approximated by the following linear model $E_{packet} = 7.96 * 10^{-6} J * s + 1.88 * 10^{-3} J$. We assume a fixed energy overhead for every packet and additional energy dependent on the packet size s.

We attribute the difference in transmission overhead to transmission power control features. Either the Cisco card or the driver automatically lowers the transmission power to the minimum required to reach the access point. Our SMC configuration in comparison does not seem to implement this and therefore has higher transmission overhead.

The higher energy/byte ratio for smaller packets can be explained with a fixed processing overhead for every packet in the DSP and MAC protocol overhead. Every packet requires that the card listens for a carrier signal with possible back-offs before transmission. In addition, it needs to receive an acknowledgement after the transmission.

Our per-packet calculation we do not consider the additional overhead of Ethernet, IP, and other headers. Thus, the per-packet overhead would be even higher if measured on application data.

6 Packet Aggregation

We believe that idle power in wireless data networks will eventually be reduced by including some sort of a "paging" mechanism, wherein a simple very low-power radio listens for a "wakeup" message that then enables the standard transceiver.[2]. Likewise, our experiments with the Cisco 350 card appear to indicate that modern 802.11b cards are including mechanisms to automatically adjust transmission power in infrastructure mode based on the signal strength or retransmission rates.

It's clear that these physical layer attributes are best handled by interface firmware; however, our measurements indicate that some energy optimizations can be performed at the link layer. Combining multiple smaller packets and transmitting them over the wireless link as a single larger packet reduces the number of packets transmitted; our measurements show this may decrease the energy needed for networking by reducing per-packet overhead and thus total power consumption.

However, it isn't clear how much packet aggregation can be performed in practice; simply counting packet distributions isn't sufficient because it would be unacceptable to significantly increase packet delay, and protocol dependences may be exacerbated by this increased delay.

We implemented a QoS-sensitive packet aggregation mechanism using the Click Modular Router. In order to increase the chances of having packets to aggregate we need to delay outgoing packets and buffer them. Packets can be no larger than the MTU of the wireless card.[3] When packets are sent by the network stack, we place them into our transmission buffer. Whenever, the buffer cannot fit the next packet, it is flushed and transmitted as a single packet. We assign a timeout to the first packet in the buffer and thereby place an upper bound on the delay for each packet. The buffer is flushed whenever it is full or the timeout occurs.

We wanted to determine if enough packets can be aggregated to reach substantial power savings and how this would increase delay. We analyzed these questions with respect to web browsing workloads, because we expect web access to be one of the basic applications of handhelds with wireless networks.

Our implementation is independent of specific applications; the packets are encapsulated and deencapsulated at the link layer. This solution requires that both communication endpoints be able to understand the encapsulation protocol. The aggregation test bed is comprised a *Dell Inspiron 3800* laptop with a Cisco 350 wireless card, an *Apple Airport*, and a workstation with an Ethernet connection to the airport and the Internet gateway. We use an infrastructure network to direct all traffic through a specific access point, insuring that our test systems would capture all traffic. Although not shown in the Click specifications below, we have implemented an ICMP extension that allows us to query remote nodes to determine if they support aggregation.

[2]This idea has been proposed by Broderson & Rabeay of the Berkeley Wireless Research Center, among others

[3]The MTU of 802.11 is larger than that for standard Ethernet, but we used the standard 1500 byte MTU to avoid problems with the network drivers.

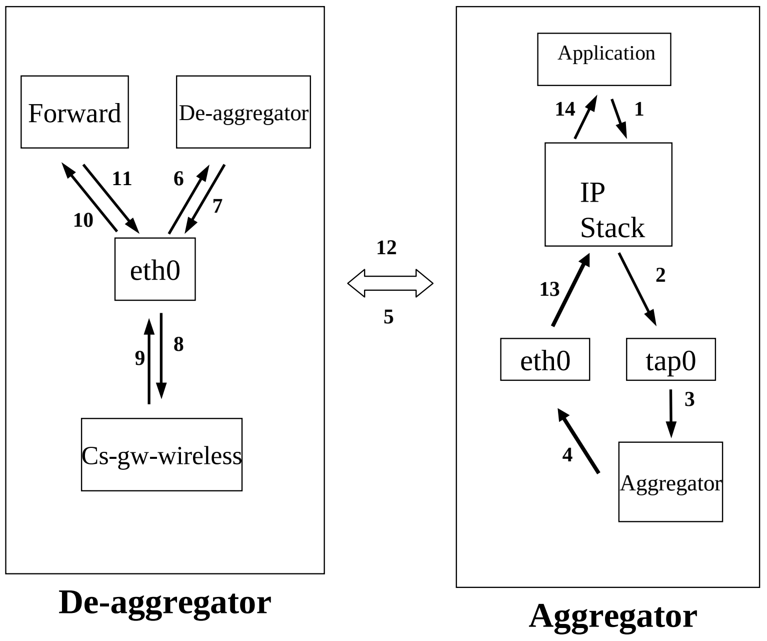

Figure 5: Aggregation Setup

We implemented an aggregation test bed to obtain realistic aggregation and delay data. We can then apply our power models from the previous experiment to calculate the energy savings. The experiment is repeated for different buffer timeout intervals. Specifically, we used the following timeouts: 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500 ms.

We implemented an aggregator and de-aggregator module in the Click Modular Router Framework [28]. These are installed on the laptop and workstation, respectively. The aggregator module performs the packet buffering and aggregation function described in the previous section. The de-aggregator extracts the original packets from an aggregated packet and forwards them to the Internet. The exact setup is described in more detail in the following subsections.

In order to get repeatable web browsing behavior, we created a script that downloads HTML files and pictures from several popular websites with the Linux *wget* program. It simulates ”user thinking” during web browsing through delays between accesses of different web pages. In addition, it mimics web browser communication behavior by downloading pictures with multiple TCP connections simultaneously.

6.1 Aggregation Setup

The software configuration of our aggregator and de-aggregator machine is illustrated in figure 5. On the aggregation machine, the aggregator module needs to intercept all packets sent by applications such as our web browser. Therefore, we install an *ethertap* virtual network interface (*tap0*) and change the default route in the IP-Stack's routing table to this interface. Then, we configured the Click router to receive packets from the *tap0* interface. Our Click module then sends the aggregated packets directly through the wireless network interface (*eth0*) to the de-aggregator machine; our implementation can also be used directly in the kernel.

The corresponding Click configuration for the aggregator machine is shown in the following code fragment. The first part routes packets from the *tap0* interface through the aggregator module to *eth0*. The second part routes received packets from *eth0* through the *tap0* interface back to the application.

```
tun :: KernelTap(192.168.0.99/25);

tun -> MarkIPHeader(14)
-> IPPrint
-> Aggregator(SIZE 1400, // maximum payload size
    TIMEOUT 100, // delay threshold
    ETHSRC 00:40:96:38:C8:52, // client
    ETHDST 00:06:5B:4A:50:E0, // deaggregator
    DST 192.168.0.1) // special IP address to indicate
```

```
                     // aggregated packet
-> Print(aggregated)
-> ToDevice(eth0);

FromDevice(eth0)
-> MarkIPHeader(14)
-> IPFilter(allow tcp)
-> Print(TCP)
-> tun;
```

The de-aggregator also performs network address translation (NAT) functions to enable the packets to pass through our local gateway. Therefore, it changes the source IP address to the de-aggregator machine and updates header checksums. Similarly, on replies from the outside network the destination address is changed to the aggregator. This part is implemented in a Click forwarding module.

Since we run Click as a user-level program, incoming packets are duplicated. They are received by the standard IP stack and by Click. In order to prevent TCP reset messages from the IP-stack before packets are forwarded (because the replies are not intended for this machine), we configure the IP stack to drop all packets via *iptables*. Since the network is only operating at 11Mbit/s, the gateway processor was able to process the full connection stream without dropping packets, even with our inefficient configuration; in measurements at MIT, the Click router has been shown to sustain wirespeed forwarding for 100Mbit/s networks.

The Click configuration for the de-aggregator is shown below. For every incoming packet we need to distinguish between an aggregated packet or a regular packet received from the outside network. For simplicity, we chose to have the aggregator set a special destination IP address to indicate that this is an aggregated packet. Aggregated packets are then split up and forwarded to the outside network. Regular packets are forwarded to the aggregation machine.

```
AddressInfo(AGG 192.168.0.1)
AddressInfo(LAPTOP 128.138.205.205)

filter::IPFilter(allow dst AGG, 1 tcp);

FromDevice(eth0)
->MarkIPHeader(14)
->Print(receive)
->filter;

// Aggregated packets: split up, perform NAT, and forward
filter[0]
->Print(allowed)
->Deaggregator(1000)
->EtherStrip(0x0800,00:06:5B:4A:50:E0,00:00:0C:0E:68:41)
->SetIPSrc(128.138.205.205)
->TCPChecksum
->Print(Stripped)
->MarkIPHeader(14)
->IPPrint
->ToDevice(eth0);

// Packets from outside network: Perform NAT and forward to aggregator machine
filter[1]
->EtherStrip(0x0800, 00:06:5B:4A:50:E0, 00:07:50:CA:89:72)
->SetIPDst(192.168.0.99)
->TCPChecksum
->Print(FILTERED)
->ToDevice(eth0)
```

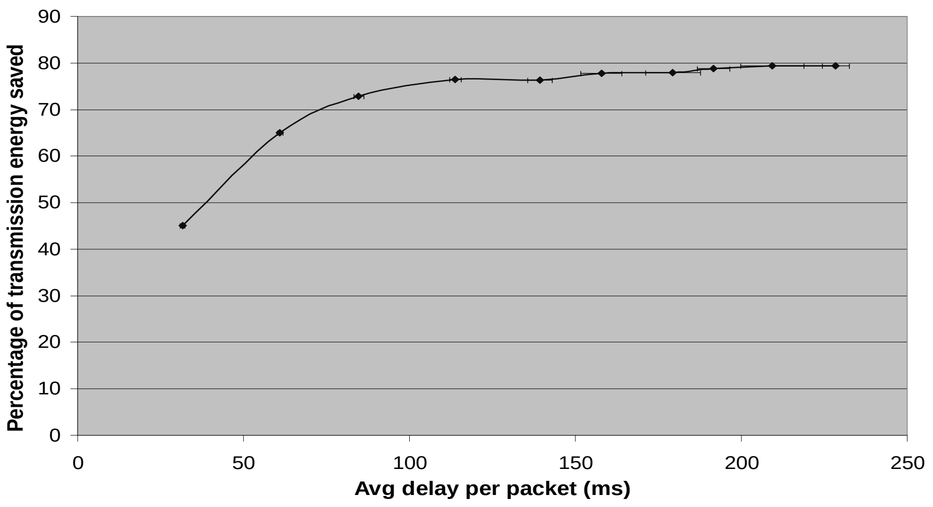

Figure 6: Aggregation results for web browsing workload

6.2 Energy Calculations

Energy and delay are calculated based on a log file from the aggregator. For every processed packet, the aggregator records the delay and whether it was combined with other packets. This gives us the number and sizes of original packets as well as the number and sizes of the aggregated packets.

We then apply our packet energy model to calculate the transmission overhead for both cases. Finally, we add the idle energy baseline for the intervals, where the card is actively transmitting. To estimate the packet processing and transmission duration, we measured the duration of a power spike for a single packet transmission on our Cisco 350 card. The results are shown in table 3. For a conservative estimate, we assumed that all original packets incurr the minimal per-packet overhead while our aggregated packets incurr the maximum per-packet overhead. In summary, the energy for every packet is calculated as the sum of transmission overhead and idle baseline for the duration of the packet transmission.

6.3 Results

Figure 6 shows the relative energy savings and average packet delays for different buffer timeout intervals. The results show that transmission energy can be cut in half at an average delay of 50ms. As expected, the energy savings increase with larger timeout values. The rate of savings levels off at about 75% with an average delay of about 100ms. It reaches a maximum of 80%.

6.4 Discussion

Packet aggregation works well with workloads where small packets are sent at a fast rate. Web browsing is such a workload. Most packets sent by the web browser are small TCP control packets such as syn, ack, or fin, or small HTTP messages. This is shown in Figure 7, where we have graphed the frequency of different packet sizes. These packets are sent at a fast rate, because the web browser opens multiple simultaneous connections to download embedded files in the web page such as pictures.

Not all workloads have the same packet distribution; for example, multimedia streaming applications typically send larger packets. This is shown in Figure 8, which captures packets from an interactive streaming audio

Packet size (byte)	50	500	1450
Duration (ms)	1.05	1.3	2.1

Table 3: Duration of power spike for single packet at 11Mbps

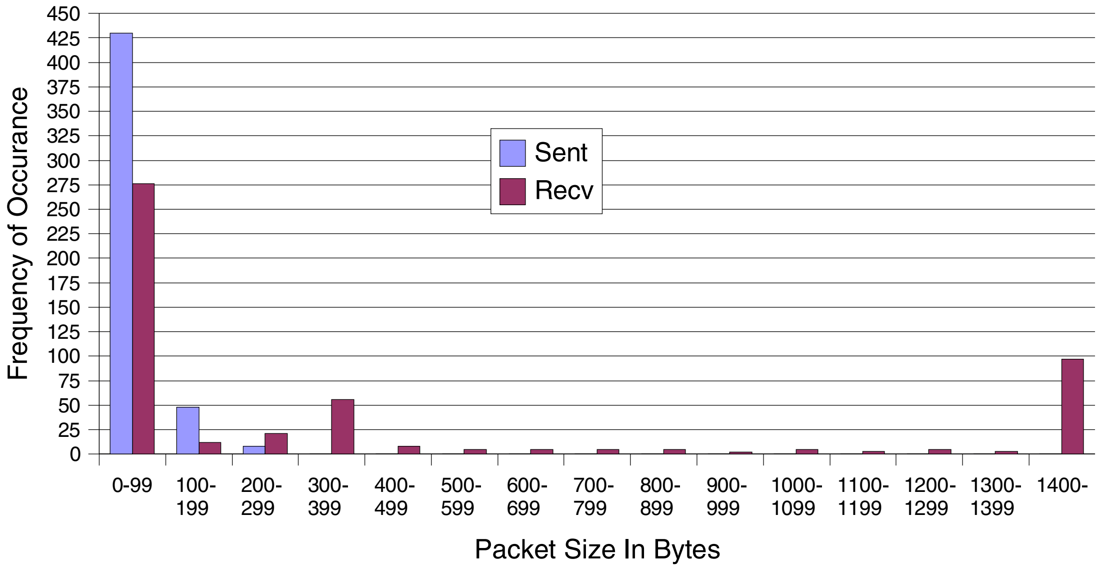

Figure 7: Packet size histogram for web browsing workload

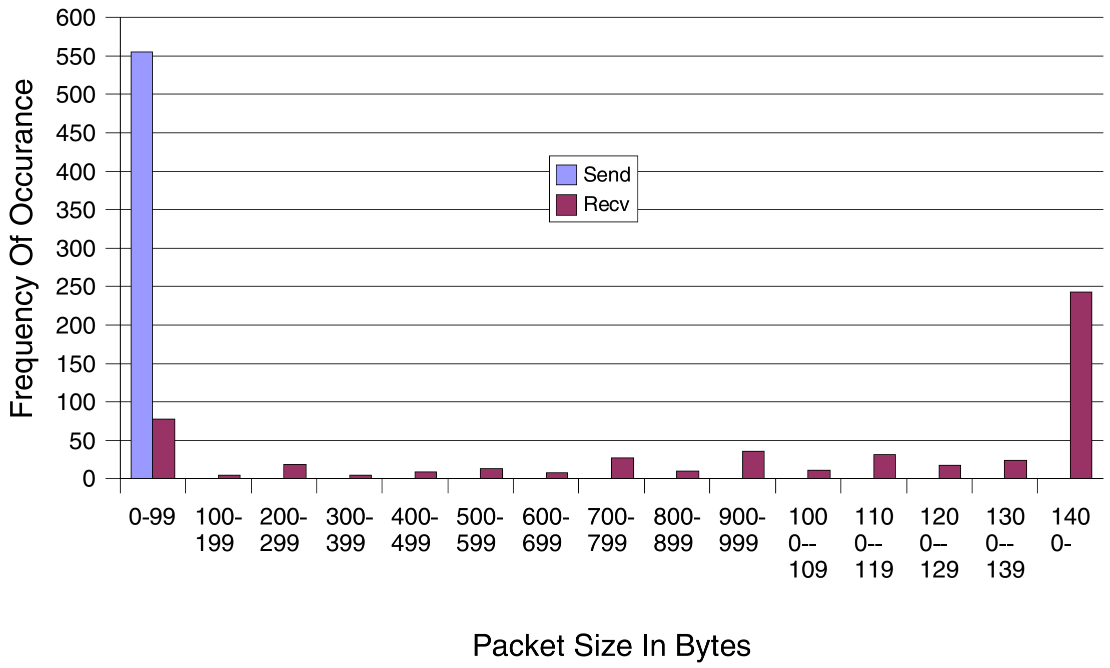

Figure 8: Packet size histogram for media workload (winamp streaming media)

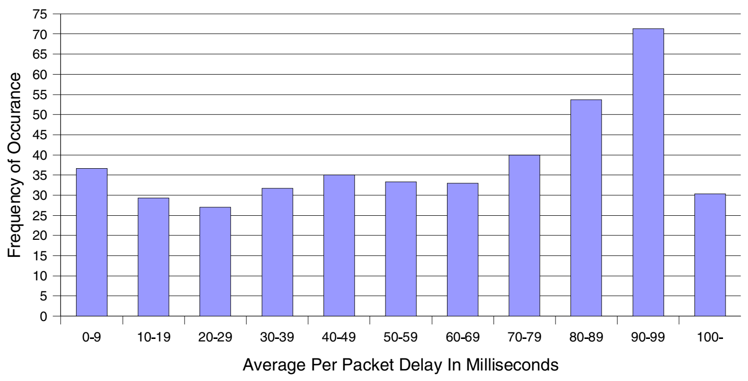

Figure 9: Distribution of packet delay when using aggreation with a delay threshold of 100 milliseconds

application (Winamp). Although this application sends larger packets, there are still many small packets (again, mainly TCP/IP ACK protocol messages).

Although each of these ACK packets is small, their frequency should still expose opportunities for energy savings. Since our packet aggrgeation module is application-independent, packets can be aggregated from multiple simultaneous applications or connections. We are starting to apply packet aggregation to another promising workload for packet aggregation, *ad hoc* routing messages, since most *ad hoc* protocols send small routing messages.

Our packet aggregation introduces a small amount of packet delay, Figure 9 shows the frequency of delay when we compared captured traces of the original workload and the aggregated workload when using a delay threshold of 100ms. Since we are using executed, but repeatable, workloads, there is some natural variation in the delay due to artifacts of the webserver and operating system overheads; however, repeated measurements show similar trends where the additional delay is evenly spread between 0 and 100ms.

Although Figure 6 only shows the decreased energy consumption of packet transmission, our results show that aggregation is applicable to both packet transmission and reception, since Figures 7 and 8 also show large numbers of small received packets. Our implementation aggregates communication in both directions. We also have to consider the energy overhead that our aggregation mechanism places on other components in the machine. Since combining packets is a simple operation, we believe that this overhead is negligible.

Finally, we point out that packet aggregation is complimentary to idle power reduction schemes. Without idle power management, packet aggregation can only save energy on the transmission overhead. Idle power management schemes benefit from packet aggregation, because the actual transmission time is reduced. Therefore, there are more idle phases, which are opportunities for power savings through the idle power manager.

7 Conclusions

Wireless network interface power consumption is significant for small mobile computing devices such as notebooks, handhelds, and sensor nodes. Especially, devices that communicate frequently cannot effectively use idle power management schemes. Therefore, transmission power control mechanisms are necessary.

This paper has presented a quantitative analysis of transmission power consumption under different bitrate and transmit power configurations. We found that existing mechanisms appear to be able to exploit physical layer energy reduction mechanisms, and focused our attention on a link layer energy reduction that cannot be implemented by the network interfaces.

We implemented packet aggregation at the link layer. It combines multiple smaller packets for transmission as a single larger packet. This method uses wireless network energy more effectively because of the high overhead for every packet. We showed that the aggregation rate is high enough for web browsing to yield up to 75% transmission energy savings when using a maximum delay threshold of 100ms per sent packet (which results in a $\approx 65ms$ observed delay). The aggregation rate is high because web browsers open multiple simultaneous connections that frequently send small control packets. We speculate that small packets are tranmsitted frequently enough for aggregation to be effective in many usage scenarios.

References

[1] R. Kravets and P. Krishnan. Application-driven power management for mobile communication. *Wireless Networks*, 6, 2000.

[2] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications, vol.E80-B, no.8, p. 1125-31*, E80-B(8):1125–31, 1997.

[3] Cisco aironet 350 series specifications. World Wide Web, `http://www.cisco.com/univercd/cc/td/doc/pcat/ao350ca.htm#spec`.

[4] Mark Weiser, Brent Welch, Alan J. Demers, and Scott Shenker. Scheduling for reduced CPU energy. In *Operating Systems Design and Implementation*, pages 13–23, 1994.

[5] Kester Li, Roger Kumpf, Paul Horton, and Thomas E. Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter*, pages 279–291, 1994.

[6] S. Singh and C.S. Raghavendra. Pamas - power aware mulit-access protocol with signalling for ad hoc networks. *ACM Computer Communications Review*, July 1998.

[7] K.M. Sivalingam, M.B. Srivastava, and P. Agrawal. Low-power link and access protocols for wireles multimedia networks. In *Proc. of IEEE Vehicular Technoology Conference*, Phoenix, AZ, May 1997.

[8] R. Prasad. *Universal wireless personal communications*. Artech House, 1998.

[9] T.S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[10] P. Gupta and P. Kumar. Capacity of wireless networks. 1999.

[11] Jeffrey Monks, Vaduvur Bharghavan, and Wen mei W. Hwu. Transmission power control for multiple access wireless packet networks. In *Proceedings of The 25th Annual IEEE Conference on Local Computer Networks (LCN 2000)*.

[12] J. Rulnick and N. Bambos. Mobile power management for wireless communication networks. *Wireless Networks*, 3, 1997.

[13] R. Kravets and P. Krishnan. Power management techniques for mobile communications. In *Proceedings of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, October 1998.

[14] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, 2001.

[15] T.X Brown. Low power wireless communication via reinforcement learning. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in NIPS 12*, pages 893–899. MIT Press, 2000.

[16] Robin Kravets, Ken Calvert, P. Krishnan, and Karsten Schwan. Adaptive variation of reliability. In *Seventh IFIP Conference on High Performance Networking (HPN'97)*, April 1997.

[17] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, J. Eric Tilton, Jason Flinn, and Kevin R Walker. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems and Principles*, Saint-Malo, France, October 1997.

[18] B. D. Noble, M. Price, and M. Satyanarayanan. A programming interface for application-aware adaptation in mobile computing. *Computing Systems*, 8(4), 1995. Also as technical report CMU-CS-95-119, School of Computer Science, Carnegie Mellon University.

[19] S. Singh and C.S. Raghavendra. Power aware routing in mobile ad hoc networks. In *Proc. of MOBICOM*, pages 181–190, 1998.

[20] Hideaki Takagi and Leonard Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, mar 1984.

[21] Volkan Rodoplu and Teresa H. Meng. Minimum energy mobile wireless networks. *IEEE JSAC*, 17(8):1333–1344, aug 1999.

[22] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MOBICOM '99*, pages 174–185. ACM, 1999.

[23] Proc. Hawaaian Int'l Conf. on Systems Science. *Energy-Efficient Communication Protocols for Wireless Microsensor Networks*, 2000.

[24] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 85–96, Rome, Italy, July 2001.

[25] Comedi. World Wide Web, `http://stm.lbl.gov/comedi/`.

[26] Ktimetrace. World Wide Web, `http://ktimetrace.sourceforge.net`.

[27] Netperf. World Wide Web, `http://www.netperf.org`.

[28] Click modular router. World Wide Web, `http://www.pdos.lcs.mit.edu/click:sosp99/`.