Exploring Adaptive Power Saving Schemes for Mobile VoIP Devices in IEEE 802.11 Networks

Shuvo Chatterjee, Dietrich Falkenthal, and Tormod Ree
Massachusetts Institute of Technology, Cambridge MA 02139
shuvo@alum.mit.edu, dlf@mit.edu, tormodin@stud.ntnu.no

***Abstract*-Current IEEE 802.11 power saving schemes provide limited savings for VoIP specific wireless traffic. This paper characterizes traffic from the two most popular VoIP service providers in the United States with hopes of developing an improved approach to save power. It proposes a novel scheme, named Adaptive Microsleep (AMS), as well as an alternative scheme named Non-Adaptive Microsleep (NAMS). Both AMS and NAMS are well suited for power saving on mobile VoIP devices by increasing the amount of time the devices spend in a low-power sleep state, but doing so without introducing additional delays that would noticeably deteriorate voice quality. Simulations show that both schemes successfully satisfy both of these primary goals, saving up to 83% power, while also meeting a secondary goal of not requiring large infrastructure changes to 802.11.**

I. INTRODUCTION

Cellular phones have become ubiquitous in society. They give users the freedom to roam freely while still maintaining connectivity. However, spotty reception is still a major drawback, especially indoors. For indoor scenarios, using commonly available WiFi connections may be an apt solution to increase coverage when regular cellular phones may be out of service range. As such, one could argue that the future of wireless communications lies in wireless VoIP phones, or at a minimum in hybrid VoIP/cellular services.

At present, cellular phones allow users 2-3 hours of continuous talk time on average. While WiFi VoIP phones seem to be the future of wireless communications, their talk time can be significantly shorter. Much of this is due to IEEE 802.11's static Power Saving Mode (PSM), which turns the antenna on the network interface card (NIC) on and off continuously at a fixed rate.

In recent years, researchers have been tackling the problem of power saving in wireless devices, and in many cases, they have created new and successful schemes. One such scheme is known as Bounded Slowdown (BSD) [1]. In BSD, the NIC adjusts its sleep time based on the network traffic. It is a successful scheme when applied to wireless Web devices. However, in this paper, we show that it is not an optimal power saving scheme for use with VoIP traffic.

Instead of using PSM or BSD, we present our own power saving scheme, known as the Adaptive Microsleep Protocol (AMS). We also present an alternative protocol, named Non-Adaptive Microsleep (NAMS). Both schemes are designed for use during VoIP conversations. Our research shows that both AMS and NAMS are better suited for VoIP traffic than PSM or BSD as they allow VoIP devices to sleep longer, thereby increasing the power saved.

II. TRAFFIC CHARACTERIZATION

To be able to define new ways of saving energy we captured traffic from two of the most popular VoIP applications in the United States, Vonage and Skype. We captured the incoming and outgoing traffic using tcpdump [2].

A. Vonage Characterization

Vonage has become the fastest growing and largest VoIP provider in the United States, reaching more than 2.2 million subscribers by the end of 2006 [3]. Vonage is not yet used on cellular phones, but could conceivably be used on a variety of mobile devices including laptops, PDAs, and WiFi phones.

We captured Vonage data on a computer connected to a hub placed between a standard Vonage compatible phone and the Vonage phone adapter. For Vonage we captured five 3 minute conversations between the Vonage phone and a Verizon network cellular phone. The captured traffic shows that Vonage uses only the UDP protocol. Fig. 1 shows a plot of the packets observed from the fourth of our five Vonage conversations. We observe from the collected data that Vonage uses a constant packet size of 172 bytes, both incoming and outgoing from the Vonage phone.

For power-saving sleep opportunities, the time between consecutive packets is more important than packet size. A plot of the aggregate cumulative distribution function (CDF) for the time between consecutive outgoing packets for all conversations is shown in Fig. 2. We observe that 98% of the packets have a spacing of less than 23ms. We also observe that only 1 percent of the packets have a delay of more than 50ms. The time between consecutive packets is concentrated at certain time intervals.

The CDF of observed time between consecutive incoming packets is plotted in Fig. 3. From the plot it can be seen that there are no such concentrations as observed with the outgoing packets. This is caused by the packets having to traverse the internet before reaching the node at which we conducted the traffic measurements. It should also be noted that 99% of the packets have spacing between 10 and 30ms.

B. Skype Characterization

We also conducted similar measurements on conversations from Skype to the same cellular phone. Skype can be used on a variety of different computer operating systems, and according to Skype's website [4] has been downloaded more than 500 million times (as of March 2007).

Second International Conference on Digital Telecommunications (ICDT'07)
0-7695-2910-0/07 $20.00 © 2007 IEEE

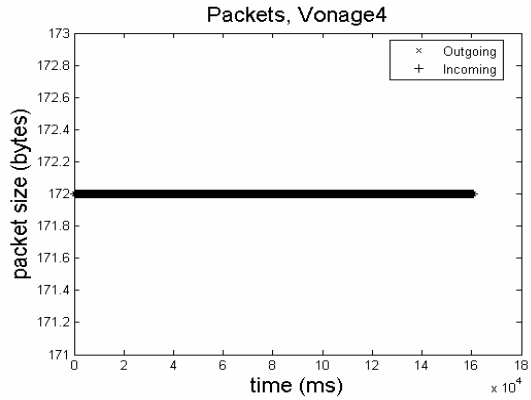

Fig. 1. Plot of a Vonage conversation (data set #4), packet size (incoming and outgoing) vs. time (in ms x 10^4)

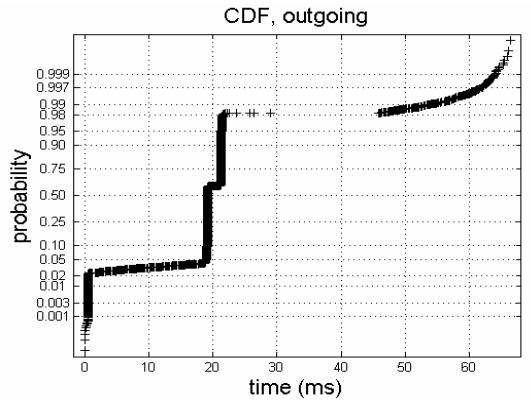

Fig. 2. Cumulative distribution function (CDF) showing time between consecutive *outgoing* packets for all sets of observed Vonage conversations.

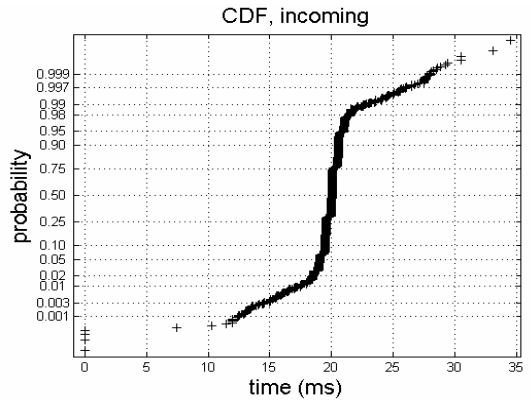

Fig. 3. CDF showing time between consecutive *incoming* packets for all sets of observed Vonage conversations.

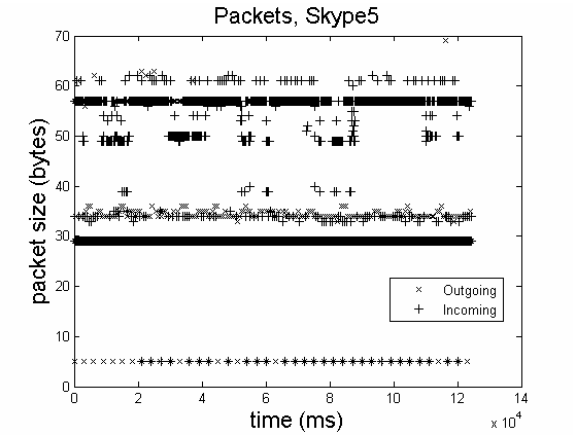

Fig. 4. Plot of a Skype conversation (#5), using UDP.

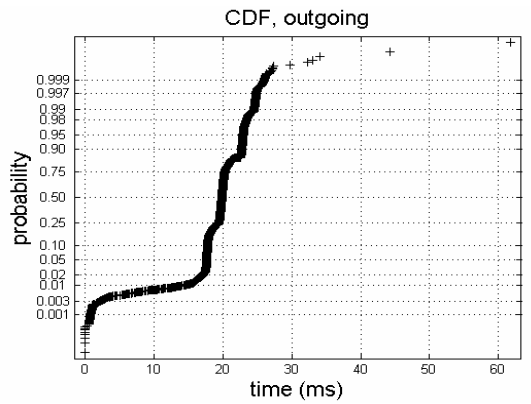

Fig. 5. CDF showing time between consecutive *outgoing* packets for all Skype convesations, using UDP, over a wireless network.

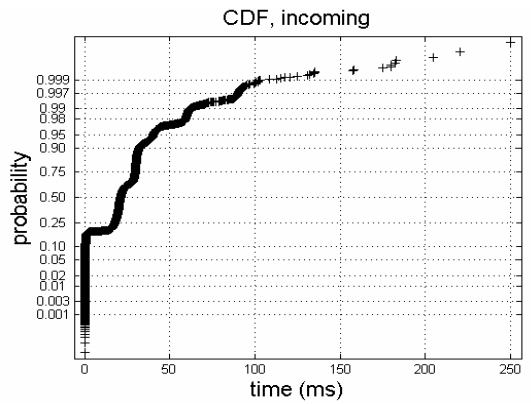

Fig. 6. CDF showing time between consecutive *incoming* packets for all Skype convesations, using UDP, over a wireless network.

For Skype, we captured traffic data for conversations transported over both the UDP and TCP protocols. Skype uses UDP to transport the conversation data as long as there are no barriers, such as firewalls, blocking UDP traffic [5]. TCP is needed for initialization. Since we are concentrating on energy savings during active conversations, the set-up phase is not included in our characterization. We conducted the measurements over a public wireless network at the Massachusetts Institute of Technology. When measuring TCP traffic, we forced Skype to utilize TCP by blocking the UDP ports. We found the TCP dumps to have more variation in packet size and delay than the UDP dumps. Because of this and because UDP is the only protocol employed by Vonage, we decided to concetrate our study on conversations using UDP. Also, UDP appears to be Skype's primary choice of transport protocol.

After initial trials, we captured ten Skype conversations over the wireless network from a laptop to the Verizon network cellular phone. The TCP ports were blocked, so only UDP was used for these conversations. We characterized the data files from tcpdump with Matlab. Fig. 4 shows a plot of one of the captured conversations. It can be seen that the packets seem to

be closely spaced. The largest packet is 69 bytes. Outgoing and incoming packets often have the same size, and the most commonly occurring sizes are 29 and 58 bytes. Skype clearly utilizes several packet sizes, while Vonage only uses one size.

For a general view of the packet spacing in Skype conversations we plotted the aggregate cumulative distribution function (CDF) for the time between consecutive packets for all our ten measurements. The CDF for outgoing packets can be seen in Fig. 5. This data shows us that 99% of the packets have spacing between 10 and 30ms.

Fig. 6 shows the same plot as Fig. 5, but for incoming packets. The distribution of time between consecutive packets is quite similar for the two plots. The major difference is that the spacing between incoming packets is more spread out than is the case for outgoing packets. We observed that more than 10 percent of the incoming packets arrive back to back. We believe that this is caused by the 802.11 Access Point (AP) buffering packets before sending them to the receiving node where we conducted our measurements. From the data it can also be seen that around 96% of the packets have spacing of less than 50ms. The incoming packets hence have a larger percentage of longer spacing than the outgoing packets.

C. Silence suppression

Jiang and Shulzrinne show in [6] that some speech encoding algorithms suppress traffic when the source is silent. We conducted experiments to find out whether such coding algorithms were used by either Skype or Vonage.

For Skype, we measured the traffic for a conversation over a wired 100Mbps Ethernet link, during a 2 minute conversation. For the first minute of the conversation, the person with the laptop muted the microphone. For the second minute, the person with the cellular phone muted the microphone. The measurements were carried out over both UDP and TCP, blocking the ports for the protocol not used. For Vonage, measurements were carried out with muting the devices in the same manner. The laptop doing the measurements was still connected to the hub between the Vonage phone and the Vonage phone adapter.

The aggregate cumulative distribution functions for the time between consecutive packets can be seen in Fig. 7. The data collected shows that there are no spacings of more than 100ms, inferring that neither Skype nor Vonage completely suppress data during periods of silence. Comparing the average time between consecutive packets for incoming and outgoing packets in this case to the study conducted earlier, it should be noted that this traffic was measured over a wired link. This removes the extra jitter introduced by the wireless link. A comparison of the muted and not muted measurements is given in Table 1. We can see that the difference in times between consecutive packets is minimal. The only significant difference is a smaller standard deviation for the incoming packets with Skype. This is most likely because the muted experiment was conducted over a wired link.

Given the small differences between the muted and not muted characterization, we conclude that Skype and Vonage do not suppress traffic during periods of silence.

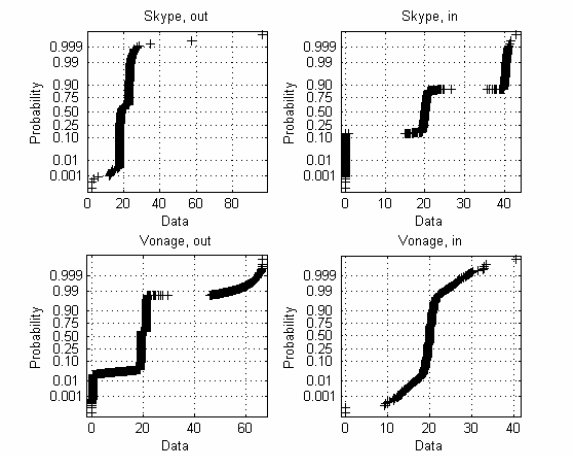

Fig. 7. Cumulative distribution function plots of time between consecutive packets for conversations with muting.

time between consecutive packets		
	mean	**st. dev.**
Vonage:		
outgoing	19.9946	5.9304
incoming	19.9938	0.9863
outgoing, muted	19.9990	5.9295
incoming, muted	19.9987	1.0984
Skype:		
outgoing, wireless	19.8599	2.1928
incoming, wireless	21.5035	14.0681
outgoing, muted, wired	20.0125	2.7448
incoming, muted, wired	19.9927	10.5297

Table 1. Mean and standard deviation for muted and not-muted Skype and Vonage conversations.

III. PROTOCOLS

D. Observations

The current protocol for power saving in an 802.11 network, Power Saving Mode (PSM), is not able to conserve power during a VoIP conversation. The interval between consecutive packets in VoIP traffic is much less than the PSM power-saving sleep interval of 100ms, thus PSM sleep mode will never kick in. Motivated by this observation, we seek a protocol that allows network cards to enter a sleep state during VoIP conversations.

Additionally, our observations show that a node using Vonage or Skype will send packets in a regular pattern with short delay between two consecutive outgoing packets. Because this delay is so short, it does not significantly delay incoming packets, nor qualitatively degrade the perceived quality of the conversation [7].

E. The AMS Protocol

The AMS protocol starts in a measuring mode, during which it determines the average time between consecutive incoming packets. This is done because different VoIP applications use different coding schemes for the voice traffic. This results in different intervals between times at which packets are sent, and thus also a difference in the inter-arrival times of packets. To conserve the maximum amount of energy, it is our goal to spend as much time as possible in the sleep state. Moreover it is also our goal to introduce as little additional delay as

possible. In an ideal situation, this would be done by predicting exactly the time of the next packet transmission or packet arrival, and entering the awake state just before this occurs. To save power while not introducing an unacceptable level of delay, we thus need to measure the average time between consecutive incoming packets. This measure is then used as a threshold for the maximum time the network interface card can spend in the sleep state. The time between consecutive outgoing packets does not need to be measured because AMS makes the NIC go from the sleep state to the awake state whenever there is a packet to transmit. Additionally, a second goal was to create a protocol that does not require significant changes to the IEEE 802.11 standard.

The time the NIC stays in the measuring mode is a static predetermined interval. After this interval has passed, the NIC enters the power saving mode. In this mode, there are two different states, sleep and awake. The protocol tries to maximize the time the NIC is in the sleep state, while constraining the delay introduced for incoming packets.

The PSM protocol requires the Access Point (AP) to buffer any incoming traffic for the node while it is in its sleep state. Traffic is then announced in a periodic beacon broadcast to all the nodes. In AMS the AP also needs to buffer any incoming traffic for a node in the sleep state. Because AMS allows the NIC to go in and out of the sleep state more frequently than PSM, we have chosen not to utilize a beacon approach to the synchronization between the AP and the NIC.

In AMS, the NIC does not notify the AP when entering the sleep state. The AP will therefore try to transmit any incoming data to the NIC. If the NIC is in the sleep state, this transmission will not be successful, and we require the AP to buffer such traffic. Since our protocol is designed to work in an 802.11 network, the absence of link layer acknowledgements will indicate a failed transmission. The AP will retransmit the buffered data to the node when it learns that the NIC has entered the awake state. This can happen in two different ways, when the AP receives data from the NIC or when it receives a polling packet discussed later.

There are two events that will make the NIC go from the sleep to the awake state. The first is if the NIC has data to transmit, which triggers an immediate transition. The other is if a maximum adaptable sleep timer, known as the sleep threshold, is reached. In the first case, the NIC will transmit the outgoing data, wait for any incoming data for a fixed period of time known as the listening threshold, and if there are no incoming packets, it will return to the sleep state. We set the listening threshold to a fixed value to allow for packets to traverse to the NIC even in the presence of contention on the network. In the second case, the NIC will transmit a packet polling the AP for any buffered data. After the polling packet is sent, the node stays awake for the listening threshold, and if there is no data to be received, it reenters the sleep state.

The sleep threshold and the polling packet are introduced to counter the situation where there is a substantial delay between two outgoing packets. In such a case, the NIC will reach the sleep threshold and enter the awake state to poll the AP for buffered data.

```
mode = measure;
measureT = preset measuring interval;
while (mode = measure)
{
   if(time()-startTime > measureT)
   {
       mode = pSave;
   }
   if((pkt received)&(mode = measure))
   {
       // update threshold with exp
       // weighted average
   }
   if ((mode = pSave)&(pkt sent))
   {
       Sleep(threshold);
   }
}
while (mode = pSave)
{
   if(wakeup&(reason = pkt_sent))
   {
       wait(listeninginterval);
       receive pkts : decrease threshold
           (multiplicative);
       no pkts: do nothing;
       sleep(threshold);
   }
   else if(wakeup&(reason = threshold))
   {
poll base station;
wait (listeninginterval);
receive pkts: decrease threshold
   (multiplicative);
no pkts: increase threshold
   (multiplicative);
sleep (threshold);
   }
}
```

Fig. 8. Pseudocode for AMS.

F. The AMS Polling Packet

The NIC polls the AP with a polling packet. The packet is a stream of 0's that is 5 bytes in length. Conceptually, the polling packet is based on 802.11's RTS/CTS frames, which are 60 bytes in length. Because under AMS the NIC polls the AP after waking every time, we are setting the polling packet to be of size 5 bytes to avoid unnecessary contention and spurious transmission over the network.

Once the AP receives the poll from the NIC, it sends the NIC any buffered packets that may be waiting. If no packets are currently buffered, it sends the NIC nothing. Likewise, the NIC waits for the listening threshold after sending the packet for any packets that might be waiting for it at the AP. If it receives no response from the AP, it determines that there are no packets for it in the AP's queue and returns to the sleep state. If it does receive packets from the AP, it stays in awake state to receive all of them. The sleep threshold is adjusted based on the response the NIC receives from the AP.

G. How the Sleep Threshold is Adjusted

After sending the AP the polling packet, if the NIC does not receive any packets during the listening threshold, it returns to sleep state and multiplicatively increases the sleep threshold by a factor of α. The idea here is that, since the NIC reached its sleep threshold the last time it was in the sleep state without ever waking to send a packet, and because no packets were waiting for it in the AP when it awoke, traffic on the link as a whole must have decreased, and therefore it can sleep longer. Conversely, if it receives data after sending the AP the polling packet, the NIC determines that it was in sleep state for too long and multiplicatively decreases its sleep threshold by a factor of β. The idea here is that, since packets were already

waiting for the NIC in the AP's queue, traffic on the link must have increased while the NIC was in the sleep state, and therefore it must quickly decrease its sleep threshold to avoid causing further delays on the link.

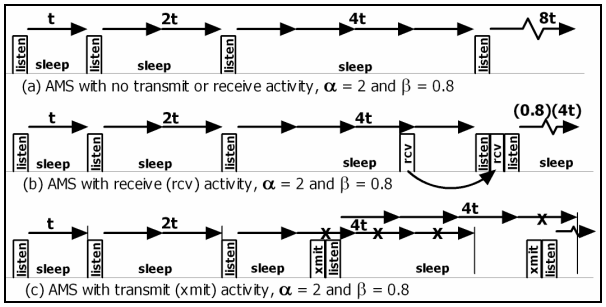

Fig. 9. Visualization of how the sleep threshold is adjusted.

H. The NAMS Protocol

We also consider an alternative protocol based on AMS. In the Non-adaptive Microsleep (NAMS) protocol, the NIC has a fixed sleep threshold of 50ms and no longer requires the use of polling. It is therefore a simpler protocol than AMS. Like AMS, however, it does require the AP to still buffer its packets. If the outgoing stream is predictable, as is the case for Vonage and Skype, then the NAMS scheme will work well. In these cases, the period between transmissions is short enough that adjusting the sleep period is unnecessary. On average, Vonage and Skype force the NIC into an awake state to send packets often enough that the NIC can receive data then, so the polling packet is unnecessary. We leave a fixed sleep threshold for instances where the device may not send a transmission for an excessive amount of time.

```
mode = pSave;
sleep;
while (mode = pSave)
{
  if (pkt_sent)
  {
      wait(listeninginterval);
      receive pkts;
      sleep(fixed_threshold);
  }
}
```
Fig. 10. Pseudocode for NAMS.

I. Comparing AMS to PSM & BSD

Our proposed protocol, Adaptive Microsleep (AMS), allows for power conservation on a much finer grained basis than PSM. Krashinsky and Balakrishnan [1] have shown through measurements that the time for a network interface card (NIC) to switch between a sleep and awake state can be less than a millisecond. AMS takes advantage of this through switching between the states more frequently than the PSM scheme allows. AMS also adapts the duration of the time the NIC stays in the sleep state to the on going traffic.

AMS and BSD are similar in that both have adaptive sleep patterns, but that is the extent of their similarities. There are several differences between the two protocols. First, AMS switches between sleep and awake state far more frequently. This allows for sleep between VoIP packets, which arrive at an interval that is less than the minimum sleep time required for BSD. Second, in scenarios where there are a large number of packets waiting at the access point for the NIC, BSD tends to drastically decrease its sleep threshold, jumping from a long sleep time to a long time in the awake state. While this technique works well for dealing with bursty Web traffic, it does not work well with VoIP traffic. Instead, AMS multiplicatively decreases its sleep threshold, which allows for smoother sleep adjustments over time. Finally, BSD utilizes the beacons already present in the 802.11 infrastructure, which means it is restricted to the beacon's 100ms constraint. Unlike BSD, AMS does not use 802.11's beacons. Instead, it utilizes its own polling mechanism for situations where the NIC reaches the sleep threshold and must let the AP know that it is in the awake state. Using the beacons in AMS would have caused too much contention for VoIP traffic.

IV. SIMULATION

Trace-driven Matlab simulations using the previously characterized Skype data files showed favorable results for both protocols. NAMS and AMS spent similar amounts of time in sleep state. Packet delay was also similar and small enough to pose little danger to degradation of voice quality. AMS performed slightly better than NAMS in limiting the maximum packet delay.

J. NAMS Simulation Runs

NAMS was fairly straightforward. We set the sleep threshold to 50ms and the listening threshold to 2ms. Once set, both parameters remained constant throughout the simulation. We arrived at the initial values by analysis and confirmed them by experimentation. A sleep threshold of 50ms was set to avoid long delays which might noticeably degrade the voice quality of the conversations. Two factors influenced our choice of the listening threshold. On one hand, a lower listening threshold gives a higher percentage of sleep time because the NIC stays awake for a shorter time. On the other hand, the listening threshold needs to be long enough to account for any congestion in the network. Congestion in the network could lead to the AP not being able to deliver pending transmissions for a newly awoken NIC fast enough to be received prior to a transition back to sleep state. After considering these factors and experimenting with different values, we decided on 2ms as our value for the listening threshold. This is the same amount of time that a NIC stays awake to listen for the beacon under the PSM scheme [1]. A more thorough evaluation of the listening threshold is difficult with data and trace-driven simulations. Implementing our protocol, however, would give new opportunities to study the real world effects of this value.

The simulation results for NAMS are shown in Table 2. We see that the NIC is able to spend 89.4% of the time in the sleep state (we discuss the energy savings that follow from this below). The average delay introduced for the incoming packets is 7.95ms. Acceptable voice quality should be obtained with this protocol, as the maximum delay introduced is 33.52ms. The cumulative distribution function for the delay introduced by NAMS is given in Figure 11.

protocol	% sleep	avg delay (ms)	max delay (ms)	# wake-up transitions per second
AMS (0.8, 1.8)	87.8%	7.90	22.46	55
AMS (0.8, 2.0)	88.1%	7.92	22.64	54
AMS (0.8, 1.2)	81.1%	7.06	19.27	85
NAMS	89.4%	7.95	33.52	50

Table 2. Simulation results for AMS and NAMS.

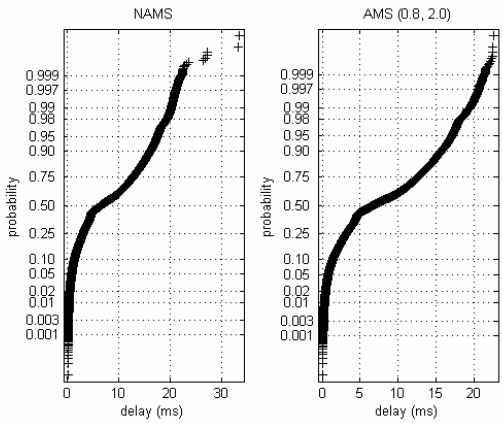

Fig. 11. CDFs for the delay time for NAMS and AMS (0.8, 2.0)

K. AMS Simulation Runs

AMS simulations were conducted in the same manner as NAMS with two additional provisions. First, we set the measuring interval length. The measuring interval is the time spent at the beginning of the conversation determining the average time between consecutive incoming packets, as described in the protocol section. Through experimentation we found 400ms provided an adequate value for the measuring interval. As in NAMS, we set the listening threshold to 2ms.

Second, values for the gain α and loss β, which during operation adaptively adjust the sleep threshold, were set. We ran simulations varying α and β, and the results are given in Table 2. Each run is denoted AMS(β, α). A more aggressive value for the increase of the sleep threshold, such as 1.8 or 2.0, give a higher percentage of time spent in the sleep state than a value like 1.2 does. We kept the value determining the decrease in the sleep threshold, β, constant at 0.8. The average and maximum delay is quite similar for the two aggressive settings. The setting with α as 1.2 gives a slightly lower average delay, and a lower maximum delay. The cumulative distribution function for the runs with β of 0.8 and α of 2.0 is given in Figure 11.

The number of wake-up transitions given in the last column of Table 2 tells us how frequently the NIC goes from the sleep state to the awake state. We see that a setting of 1.2 for α causes more transitions than for the other settings. This is caused by the NIC reaching the sleep threshold more often.

Comparing the AMS and NAMS results, we see that NAMS is able to spend a larger percentage of the time in the sleep state. It does, however, have a slightly longer average delay and much longer maximum delay.

L. Power Savings

From the simulation results, we calculated the energy savings for two different NICs. The energy spent sending and receiving data was negligible compared to the energy spent listening and sleeping by comparing the energy consumed in these situations. For example, the energy spent transmitting a 58 byte packet, including headers, over a 5Mbps link using the power specifications of the Aironet 350 NIC card is 0.28mJ. The energy consumption for listening for 2ms using the same specifications is 1580.94mJ. Since the energy spent listening is several orders of magnitude larger than the energy consumed during transmission, we safely disregard the latter's energy. Like Krashinsky [1], we also determined that the small spike in power consumption observed with the Enterasys RoamAbout NIC could be ignored due to its short duration. This spike can be seen in Figure 12.

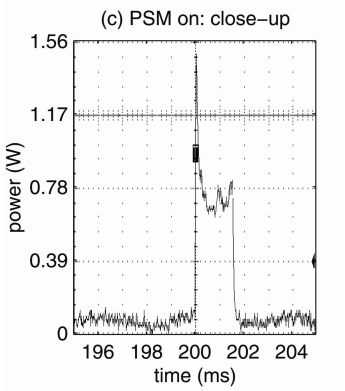

Fig. 12. Power consumption of the Enterasys RoamAbout NIC, in [1].

We calculated the energy savings by using the power measurements shown in Table 3. The results (Table 4) show the calculated power savings each protocol could achieve using these two different NICs. We arrived at the energy savings by combining the time spent in the sleep and awake state with the listen and sleep power levels.

phase	power	
	Aironet 350 NIC	Enterasys Networks RoamAbout NIC
listen	790 mW	750 mW
sleep	169 mW	50 mW
transmit	1304 mW	-
receive	955 mW	-

Table 3. Power consumption for Aironet 350 and Enterasys RoamAbout.

protocol	energy savings	
	Aironet 350 NIC	Enterasys Network RoamAbout NIC
AMS (0.8, 1.8)	69.02%	81.95%
AMS (0.8, 2.0)	69.25%	82.23%
AMS (0.8, 1.2)	63.75%	75.69%
NAMS	70.28%	83.44%

Table 4. Energy Savings

We see from the results in Table 4 that the Enterasys card has higher power saving than the Aironet card. This is simply an attribute of the lower sleep power level for the Enterasys. When looking at the AMS power savings, we see that the

more aggressive settings for α of 1.8 and 2.0 give slightly higher energy savings than the more passive setting of 1.2. Our results also show that NAMS gives a higher energy saving for both NICs than any of the AMS runs.

M. Implementation

One of the design goals for our protocol was that it should be easy to implement in an IEEE 802.11 wireless network [8]. For this reason, we tried to reduce the number of changes that would be needed to the firmware and physical properties of the network components.

Both AMS and NAMS require small changes to the behavior of the current network elements. None of them require changes to the physical properties. AMS requires the AP to buffer incoming traffic that it is not able to transmit to the sleeping NIC. Current APs buffer traffic for nodes sleeping under the PSM scheme [1]. Under AMS, however, the AP will not know which NICs are sleeping, and must therefore first try to transmit the data. If the transmission is unsuccessful, the data must be buffered. The other minor change required under AMS is that the AP must send any buffered data when it receives indication that the NIC has left the sleep state. This will be indicated by either the reception of a regular packet or a special polling packet. NAMS requires somewhat smaller changes to the AP. With NAMS, the AP does not have to recognize the polling packet, since it is not used with this protocol. The other changes required are the same as for AMS.

We believe that the end nodes implementing AMS and NAMS will be mobile handsets tailored for VoIP communication over wireless networks. Current NICs can be adapted to AMS and NAMS through changing the firmware of these cards. The cost of doing so could be minimal compared to the potential power savings.

V. RELATED WORK

This paper has two contributions. First, it characterizes data gathered during VoIP conversations with the two most popular VoIP providers in the United States. We show that current power saving schemes, such as PSM and BSD [1], are not suitable to save energy with such data patterns. Second, we propose two protocols, AMS and NAMS, that we believe are able to significantly reduce the power consumption during VoIP conversations in IEEE 802.11 networks. Our trace-driven simulation results show energy savings of up to 83%.

A survey of power saving schemes for wireless networks in different protocol layers is given in [9]. Krashinsky and Balakrishnan propose an adaptive sleep algorithm for wireless web access, BSD, in [1]. This approach shows some similarity to ours by adapting the time the NIC sleeps. There are however, several major differences, as we discussed in section 3.6. Kravets and Krishnan propose a power saving scheme involving sleep in [10]. This approach introduces delays of 0.3-3.1 seconds and is controlled at the transport layer. It is thus not suitable for delay-sensitive traffic such as voice conversations. Another approach to power management is to control the power at which the transmissions are done. Qiao *et. al.* propose such a scheme in [11]. When dealing with voice transmissions it is important to consider the voice quality of these. Kotwicki analyses the effects of loss rate and delay in [7]. There have also been numerous papers on power saving in sensor networks, such as [12, 13].

VI. CONCLUSION

In this paper, we successfully characterized traffic from the two most popular VoIP applications in the United States. We also proposed the Adaptive Microsleep (AMS) and Non-Adaptive Microsleep (NAMS) protocols, two power schemes designed to be used during VoIP calls on wireless mobile devices. Using simulations, we found that our protocols have power savings of up to 83%. We accomplished this by increasing the sleep time for the network interface card, and we did so without introducing any significant delays. Additionally, our protocols do not require large infrastructure changes to 802.11. We further conclude that the NAMS protocol may be better than AMS with traffic such as that from Skype and Vonage, since it is a simpler protocol. Further work includes hardware implementation and testing as well as research to confirm AMS and NAMS to be generalizable beyond our test cases.

ACKNOWLEDGMENT

We would like to acknowledge Ronny Krashinsky and Prof. Hari Balakrishnan for helpful comments and feedback. We would also like to thank Leo Dertouzos for use of his telephone test equipment.

REFERENCES

[1] R. Krashinsky and H. Balakrishnan. *Minimizing Energy for Wireless Web Access with Bounded Slowdown*. Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, September 2002.

[2] The tcpdump/libcap project. www.tcpdump.org.

[3] Vonage, http://ir.vonage.com/faq.cfm?FAQID=2, accessed 4/18/2007

[4] http://share.skype.com/sites/en/2007/03/half_a_billion_downloads.html, accessed 4/18/07

[5] Salman A. Baset and Henning Shulzrinne. *An Analysis of the Skype Peer-to-Peer Internet Telephone Protocol*. Columbia University, September 2004.

[6] Wenyu Jiang and Henning Shulzrinne. *Analysis of On-Off Patterns in VoIP and Their Effect on Voice Traffic Aggregation*. In The 9th IEEE International Conference on Computer Communication Networks, 2000.

[7] J. Kotwiki. *An Analysis of Energy Efficient Voice Over IP Communication in Wireless Networks*. Master's Thesis, Case Western Reserve University. March, 2004.

[8] IEEE Computer Society LAN MAN Standards Committee. *IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. August 1999.

[9] Christine E. Jones, et. al. *A Survey of Energy Efficient Network Protocols for Wireless Networks*. Wireless Networks 7 (4) (2001) 343-358.

[10] Robin Kravets, P. Krishnan. *Power Management Techniques for Mobile Communication*. ACM Press, 1998.

[11] Daji Qiao, Sunghyun Choi, Amit Jain, Kang G. Shin. *MiSer: An Optimal Low-Energy Transmission Strategy for IEEE 802.11 a/h*. Mobicom, September 2003.

[12] Wei Ye, John Heidemann, Deborah Estrin. *An Energy-Efficient MAC protocol for Wireless Sensor Networks*. Proceedings of the IEEE Infocom, 2002.

[13] Yeonkwon Jeong, J.P., Joongsoo Ma, and Daeyoung Kim. *An Enhanced Power Save Mode for IEEE 802.11 Station in Ad Hoc Networks. 2004*. Daejeon, Korea. pp. 414-420.