Energy Efficient Data Filtering and Communication in Mobile Wireless Computing

Tomasz Imielinski, Monish Gupta, Sarma Peyyeti
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903
{*imielins,monish,peyyeti*}*@cs.rutgers.edu*

Abstract

Battery power is a scarce resource in mobile wireless computers. We discuss energy-efficient software solutions for wireless information services and general data communication. Our approach is based on allowing the receiver to be shut off when no communication is taking place and letting the CPU be in an energy efficient *doze mode* as long as possible. Minimizing transmissions (which are much more energy consuming than receiving) from mobile hosts also leads to efficient use of battery power.

We present two protocols based on this approach. The first protocol uses multicast addresses to filter out unwanted packets without waking up the CPU. Reusability of local addresses, similar to frequency reuse, facilitates using a limited multicast address space. The second protocol advocates a wake-up based scheme to save receiver power in a mobile host. We discuss the tradeoff between energy savings and increase in delay.

Experiments with implementation of these two protocols show that a significant improvement in battery usage can be achieved, making a case for certain features to be provided in hardware.

1 Introduction

Energy is a major bottleneck for mobile computers. With the very slow expected growth rate of battery life (only 20% in the next 10 years [12]) energy efficiency is a necessary feature, both on hardware and software levels. Hardware providers are beginning to offer energy efficient solutions by allowing the terminal to switch off the background light of the screen, slowing down the rotation speed of the disk (or eliminating it completely in favor of the flash cards) and offering CPUs with energy efficient *doze mode*. For example, the Hobbit processor from AT&T consumes 5,000 times less energy while in the doze mode than in the active mode (250 milliwatts in the active mode as opposed to 50 microwatts in the doze mode).

There is a growing pressure on software vendors to offer energy efficient software solutions. In this paper we provide energy efficient solutions to data filtering and subsequently generalize it to general data communication. We will assume that the mobile client is equipped with an Ethernet card (or a similar network interface device), its CPU has an energy efficient doze mode and finally, that the mobile client has the ability to switch off the receiver (this will be necessary only for the second protocol covered in section 3). We believe that such a feature, if not yet provided, will have to be offered for mobile computers[1]. In fact this paper can be treated as an argument to convince hardware providers to offer such features. The ability to switch off the receiver extended the battery life of cordless phones from a few hours to a week [4] and is also used in some pagers, which shut their receivers off periodically. A cordless phone listens to a signal from its base station only at predefined time slots and *goes to sleep* otherwise. The delay in receiving a call is not noticeable for the user at all.

The WaveLAN/PCMCIA card (a network adapter card providing wireless communication in a PC based LAN) draws approximately 3.4 Watts for transmitting and 1.8 Watts for receiving. We will show how switching off the receiver along with keeping the CPU in a doze mode could lead to dramatic energy savings.

We will consider the following two scenarios:

[1] Currently, the receiver off feature is not usually provided

- Data Filtering

  The mobile user is often a member of various multicast groups and occasionally receives large pieces of data such as news, stock quotes etc. The user is usually interested only in a small fraction of these news. For example, one may only be interested in an IBM quote, out of the whole multicast of a set of quotes for, say, 1000 companies. Since each packet of the stock multicast has the same multicast address, the software has to look at each packet of such multicast in order to filter out the IBM packet. Thus, CPU has to be active throughout the duration of the multicast. This is a waste of energy.

  We provide a method in which the MSS (Mobile Support Station) provides a much finer granularity of indexing which enables the clients to selectively listen only to a small fraction of the news. This allows the CPU to remain in the doze mode most of the time and leads to significant energy savings.

- Data Communication

  This is a more general paradigm. Currently, the client's receiver has to be *on* even if there are no messages directed to that client. We would like to be able to allow the receiver to be shut off when no messages are expected to be received by that client. This has to be fully coordinated with the MSS, analogously to the way the cordless phone synchronizes itself with its base station. Each message sent by the MSS to a client has to be first signalled by sending a wake up message during the predefined slot. Further in the paper we show that this leads to significant energy savings.

We will discuss the two concepts in the next two sections. In general the following basic modes involving the receiver and the CPU will be considered :

- Level 0: Both the receiver and the CPU are *on*
- Level 1: The receiver is *on* and the CPU is in the doze mode
- Level 2: The receiver is *off* but the CPU is *on*
- Level 3: The receiver is *off* and the CPU is in the doze mode

Higher levels reflect higher energy savings. We assume here that the CPU in the active mode draws lesser energy than the receiver. If this assumption does not hold then Level 1 is more energy efficient than Level 2. In any event, Level 3 is the most energy efficient.

We proceed first with the description of the Level 1 solution which is used for energy efficient data filtering. The Level 3 proposal is described next.

2 Receiver On and CPU in a Doze Mode: Level 1

Here we describe how we can accomplish energy efficient data filtering using Level 1 features. As in the model of Fig. 1, Server is a host on the fixed network, MSS is a fixed host with a wireless interface and MH is a mobile host communicating with the MSS over a wireless channel.

2.1 Information Dissemination in a Wireless Environment

We consider several possible strategies for information dissemination over a wireless cell. In all of the cases it is assumed that the Server to MSS communication is through periodic multicasting of data at a higher granularity. For instance, all Stock data is periodically multicasted to a *single* group address over the fixed network. This is a reasonable assumption for any scalable solution for disseminating information over a wide area internetwork.

We envision two possible different multicasting services for information dissemination. The same data can be sent using *Regular* service and the *Refined* service. Regular service provides no filtering at the MSS which simply forwards each multicast over its wireless cell. In the refined service the client is offered the ability of lower granularity filtering as well as secondary query processing, which is discussed later in this section.

1. Periodic multicasting from MSS to mobile hosts with Application Level Filtering at the mobile. Here, the MSS simply forwards the multicast it receives from the server.
2. On demand service. The mobile host transmits its query uplink and MSS replies using the data last multicast by the server. Both, the query and result are unicast (point to point) communications.
3. Mobile host transmits uplink only changes in its query status. Thus, the MSS stores the queries

for each mobile host in its cell. On receiving a new multicast from the server the MSS forwards the query results for each mobile host over the wireless cell (results can be either unicasted or multicasted).

4. Periodic multicasting from MSS to mobile hosts with Hardware Level Filtering at the mobile. This is discussed in the next subsection.

2.2 Hardware Level Filtering

This approach is based on the publisher-subscriber model of [11] with the critical difference that filtering is done at the network interface level rather than in software, enabling the CPU to stay in doze mode for longer periods of time. The server is responsible for periodically multicasting information (which can be newsgroups, stock values, etc.) over the network.

Assuming that the server gives only one general multicast address for a large collection of packets it requires that the MSS "repackage" or encapsulate the incoming multicast stream by giving smaller pieces of information separate multicast addresses.

We assume that the multicast address space will consist of permanent and transient addresses. Permanent addresses are network wide addresses allocated statically, like the network addresses of hosts on the Internet. Transient addresses are assigned from a pool of unallocated addresses. An MSS providing a refined service would choose a set of addresses from this transient address space. The client, upon joining such a refined service, will receive a form listing predefined multicast addresses as well as a hashing function for hashing queries to multicast addresses.

This hashing function is used at the MSS to map different packets to different multicast addresses depending on a primary key. The same hashing function is used at the mobile host to map primary queries to multicast addresses. The local addresses can be reused across different cells, perhaps for different services. This reusability is helpful when we have a limited number of multicast addresses.

The discussion so far applies to any information dissemination service. In the next subsection we describe an implementation of one such service, that of multicasting stock quotes. This implementation is based on hardware level filtering. Although we describe the implementation details for the stock example, the model is general and applies to any such service. Later we discuss how this model would apply to publishing newspaper columns.

2.3 Example Application - Broadcasting a Stock Database

A Stock Server running on a host in the Fixed Network periodically multicasts the stock database on a single multicast stock group address. All the hosts functioning as an MSS (base station) join the Stock Group if they are to provide the stock service to mobile hosts in their cells. Mobile hosts run client programs that provide a graphical interface for the user to query the stock database and to view the results of these queries. Fig. 1 shows the overall model for this application.

Below we describe the operation of the server, MSS and the mobile client. User queries are based on the stock's ticker tape symbol, and various descriptors and their combinations. Simplest queries involve just recent quotes of a particular stock. These are handled by using a hashing address corresponding to the name of the stock. Such queries are called *primary queries* because they deal with primary keys. More complex queries deal with *secondary* descriptors of stocks dealing with other attributes (secondary keys) (such as P/E ratio etc) as well as with new events such as stock value reaching a new high etc. Essentially, each query is a conjunction of "$\langle Attribute \rangle = \langle Value \rangle$" conditions, where "$\langle Attribute \rangle$" can be the stock ticker symbol (a primary key) or any other secondary key (such as stock's P/E ratio etc).

Server Periodically sends all the stock packets followed by ⟨EndOfBroadcast⟩ to a well known multicast address G (Stock Group)

The **MSS** performs the following actions repeatedly

- Listen for multicasts on the Stock Group.
- Take each packet of the multicast and change its multicasting address applying the hashing function h to the stock quote contained in the packet. Store all the packets until an End of Broadcast is received.
- Form the 'Index Packets' (described later) for each descriptor using the stored packets and send them on well known Multicast Addresses over the wireless cell.
- After some delay to ensure that mobile hosts have joined the required groups, send the stored

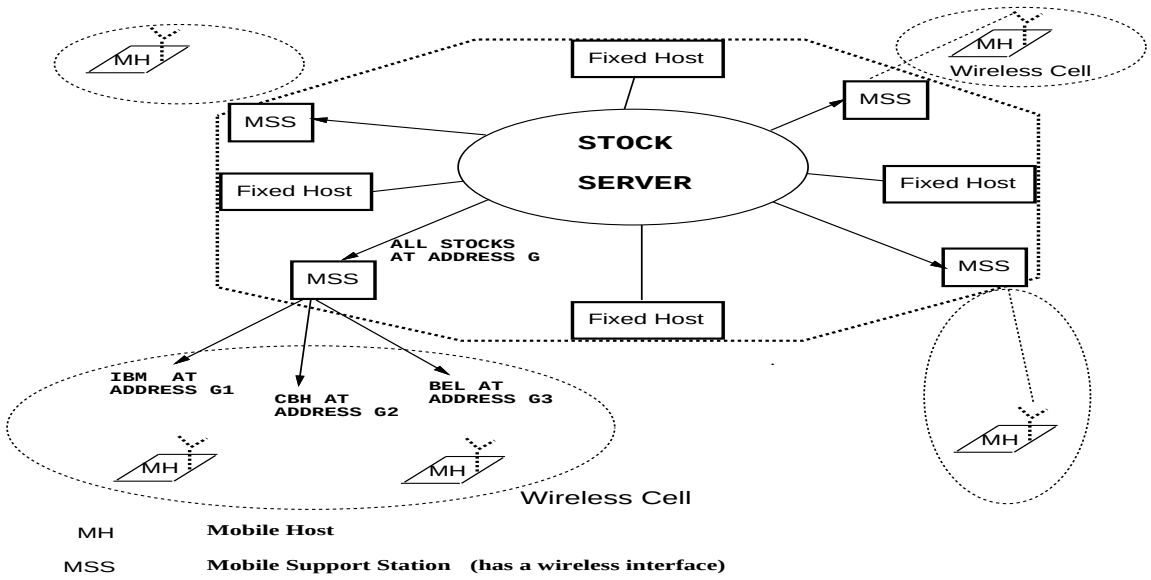

Figure 1: Model for broadcasting stock information

packets to their respective addresses. A delay of 1 second was enough in our wireless LAN environment.

- Send ⟨EndOfBroadcast⟩ on a well known multicast address.

The **Mobile-Host** performs the following actions

- Join multicast group $h(K)$ where K is a stock of interest to the user. Also join the groups for all the Index Packets that are required for the secondary queries.
- For each multicast from MSS, first get all the index packets that are required. Join the relevant addresses in the index packets.
- Read all the stock packets that arrive for the groups joined until an ⟨EndOfBroadcast⟩ is received on a well known group address.

Due to possible hashing function collisions it is still possible that the client will get a "false alarm" and the CPU will have to be woken up to conclude that after all the packet was irrelevant.

2.3.1 Hashing Function

In our implementation of the stock application, the first three characters of the ticker tape symbol are hashed into the address space of Multicast IP addresses. Multicast IP addresses are 32-bit long addresses beginning with the bit sequence 1110. Thus, a total of 2^{28} Multicast IP addresses are possible. We assumed a restricted space of the first 10 bits for the refined stock service and hash the first 3 characters of the ticker tape symbol into these 10 bits using mid-square hashing.

2.3.2 Secondary Queries

Queries based on Events and Descriptors (such as Stock reaching New High, a New Stock, etc.) are more difficult to handle. Answer to such a query returns a set of records instead of a single record. Since these descriptors are just boolean flags in the stock objects, a simple hashing scheme cannot be used, as in the case of primary queries.

Our solution is to precede the multicast of the entire stock ticker tape with a set of index packets. One index packet is sent for each descriptor. This packet contains a list of pointers - multicast addresses on which stock objects satisfying that descriptor can be received. For instance, an index node for "New High" contains a list of multicast addresses for stocks which reached a new High value.

On the client side, a query like "All objects satisfying New High " corresponds to joining all addresses in the index packet for New High. ANDing and ORing of descriptors can be done by taking intersection and union of the corresponding set of addresses. Also queries like "IBM objects with New Low" correspond to joining the address X where $< IBM, X >$ is a tuple in the New Low index packet. The index

packets are sent on well known multicast addresses. A mobile host joins the groups for all the index nodes that it needs to process its queries. Fig. 2 shows an example of index packets. With this set of indexes, a mobile host interested in all Apple stocks would join the multicast address 225.6.250.1, someone interested in all stocks that reached a New High would join 225.7.1.200, 225.15.4.120 and 225.7.2.10. The query "Stocks that reached a New Low and were Split" corresponds to 225.10.0.1. The query "Microsoft stock if it reaches a New Low" will not cause the mobile host to join any groups.

2.3.3 Configuration

We have implemented the above service using Multicast-IP [3] over Ethernet, with a Wavelan network providing the wireless cell. Local addresses are used by sending IP datagrams over the wireless with a TTL value of 1. The Server runs on a SunSparcstation running SunOS, mobile hosts are laptop computers communicating with MSS using PCMCIA card. MSS is an Intel 80486 based PC-AT running Mach with a Unix server. MSS has a wavelan wireless interface and a wired Ethernet interface.

Currently the service is implemented over a single wireless cell. To incorporate host mobility between cells a *greeting* protocol needs to be implemented for the MSS to inform the mobile host about the address space and hash function for the stock service in its cell.

2.4 Another Example Application - Newspaper Publishing

Newspaper publishing is another example service which can use our protocol for energy savings. The essential difference with stock application is that the query result size is usually bigger (of the order of Kbytes). Example queries are "Getting a business column of NewYork Times", "Getting headlines of today's news" etc. Since information here is organized hierarchically, the user can issue queries based upon the results of previous queries while this is not the case in stock application.For example, the user can selectively request "hot" news items from headlines of the news.

2.5 How much energy can we save

As we said before, the energy saving is due to the client's ability to stay in a doze mode and wake up only when information which is directly relevant is multicasted. Also, due to periodic multicasting, no transmissions are needed at the mobile hosts. Transmitting being much more energy consuming than receiving, this is another reason this protocol will be energy efficient compared to conventional, demand based information delivery.

Let P_active, P_doze be the CPU power specifications for the CPU in active mode and doze mode respectively.

Let R_active, $R_mismatch$ be the power consumed by the receiver for receiving a packet and for filtering out a packet (due to address mismatch) respectively. T_active is the power consumed in transmitting.

Usually, $P_active \gg P_doze$ and $T_active > R_active$.
We assume that $R_active \gg R_mismatch$, i.e. power required to detect that a packet is not destined for us is relatively negligible, as only the address portion of the packet needs to be read.

Further, let
N be the size of database being multicasted in number of packets,
BW be the bandwidth over the wireless Cell (in bytes/sec),
P be the size of each packet (in bytes),
R be the average number of packets a user is delivered as query results,
Q be the average query size (in bytes) generated by each client between successive multicasts.
X be the average time CPU needs to be in the active state to transfer one byte of data between the application software and the network interface.

1. Application Level Filtering at the mobile.

   CPU Energy consumed $= (N*P*X)*P_active$

   Receiver Energy consumed $= (N * P/BW) * R_active$

2. On demand service.

   CPU Energy consumed $= (R*P*X)*P_active + (Q * X) * P_active$

   Receiver Energy consumed $= (R * P/BW) * R_active$

   Transmitter Energy consumed $= (Q/BW) * T_active$

3. Storing queries at MSS.

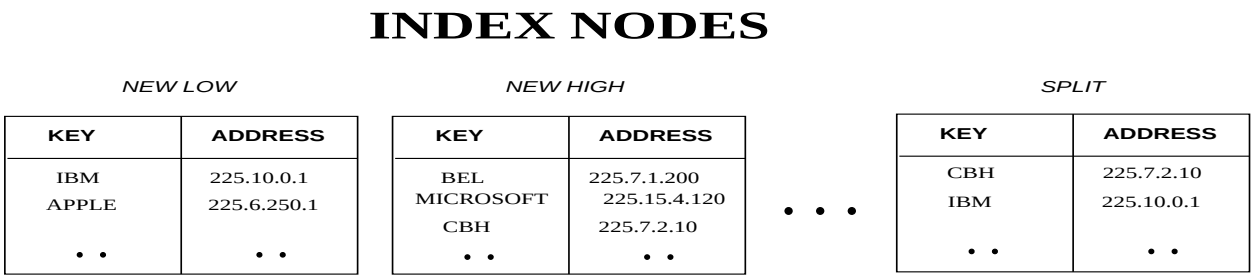

Figure 2: Index Packets

Let F be the average fractional change in queries at each client in time T. So, $F*Q$ bytes are transmitted by a mobile host to inform the MSS of its change in queries.

CPU Energy consumed = $(R*P*X)*P_active+(F*Q*X)*P_active$

Receiver Energy consumed = $(R*P/BW)*R_active$

Transmitter Energy consumed = $(F*Q/BW)*T_active$

4. Hardware Level Filtering at the mobile.

   Let L be the average number of overhead packets. These are the index packets received and packets received due to collisions. These collisions could be due to the hashing function used in the application or due to collisions occurring in translating multicast addresses (Link Level and IP Level Collisions [3]).

   CPU Energy consumed = $((R+L)*P*X)*P_active$

   Receiver Energy consumed = $((R+L)*P/BW)*R_active+((N-R-L)*P/BW)*R_mismatch$

Schemes (2),(3) are demand-based while (1),(4) are based on periodic multicasting. (3) is obviously more energy efficient than (2) and (4) is more energy efficient compared to (1). Assuming $R_mismatch \ll R_active$ and $P_doze \ll P_active$, it can be derived that (4) is more energy efficient than (3) if $L/(F*Q) < (1/P)*\frac{P_active*X+T_active/BW}{P_active*X+R_active/BW}$. Thus, for larger collision rates (3) would be better than (4), whereas (4) is more energy efficient for large values of F. For applications such as newspaper browsing frequent changes in queries will be a common scenario. For large cell sizes, higher $\frac{T_active}{R_active}$ ratio is in favor of (4) being more energy efficient than (3).

Also, in (3) queries for all the hosts are stored at the MSS. When a mobile host changes cell, this state information will either have to be transferred to the new MSS or the mobile will need to transmit all its queries uplink to the new MSS.

2.6 Conclusion

To summarize, we have achieved the level 1 of power saving by providing a more refined data filtering to the clients on the *network level*. This allowed us to keep the CPU in a doze mode and use the higher levels of the protocol stack only in case when the packet relevant to the user's query is received. We have demonstrated how multicast addresses used for the finer granularity indexing can be reused in different cells by doing encapsulation of data at the MSS.

3 Receiver Off and CPU in the Doze Mode: Level 3

In this section we show how level 3 energy saving mode can be implemented for applications which do not require a real time response, and can tolerate some delay. E-mail and news group reading belong to this category. The ideas here are a direct generalization of the previously mentioned energy efficient features used in cordless phones.

3.1 Preview Edition Protocol

The communication channel between the MSS and a Mobile Host is assumed to have the logical layout depicted in Fig. 3

Each mobile host keeps its receiver on during the *preview* period. MSS constructs a *wake-up* packet during the preview and broadcasts it to all mobile hosts in its cell. The wake-up packet contains a list of addresses of those hosts for which messages will

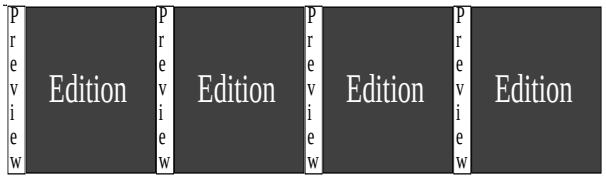

Figure 3: Preview Edition Cycles

be sent during the following *edition* period. The mobile host keeps its receiver on only if it receives the wake-up packet and finds its address in the list. Otherwise, it turns the receiver off until the next preview period.

For each message that the MSS needs to send to a host, it checks if the wake-up packet for the current preview-edition cycle has been sent and if it contained the host's address. If so, the message is forwarded immediately, else it is buffered until the next wake-up packet has to be sent. This necessitates any receivers that were left *on* to stay *on* for the entire edition period.

In case there are messages to be multicasted to a group, the MSS adds the corresponding multicast addresses in the wake-up packet. The mobile host not only checks for its own address but also the multicast addresses for the groups to which it belongs.

Let *layout ratio* r be defined as $r =$

(size of preview period)/(size of edition period)

and *activity ratio* A be defined as $A =$

fraction of time the receiver is *on*

Clearly, lower the activity ratio, higher is the energy saving obtained from the preview-edition protocol. If the mobile host receives no messages, it can keep the receiver off for $1/(1+r)$ fraction of total time. So, $A \leq 1/(1+r)$. This leads to very significant energy savings as compared to the current situation when such a host would have to keep the receiver on, leading to unnecessary waste of energy.

Intuitively, longer edition time leads to longer average delay of message delivery. For instance, if the edition time is 9 minutes, the message received by the MSS, while it has already started dispatching messages in the current edition, can be delayed 9 minutes and possibly more. The same layout ratio r can be implemented in many different ways ranging from very short preview and edition sizes to much longer previews and editions. For example, the layout ratio $r = 0.1$ could be implemented by the preview of 1 second and the edition time of 10 seconds, as well as by the preview time of 1 minute and the edition time of 10 minutes. The latter solution leads to much longer delays per message but has an advantage of the longer preview time. Such longer preview time minimizes the probability of the client losing the wake up call[2].

Below we describe the experimental results in which we varied the layout ratio, message volume and the sizes of the preview, edition periods. We have measured the average delay and the activity ratio.

3.2 Experiment

The experiments have been performed on an Ethernet based LAN. The base station and the client (mobile host) are simulated as two processes on different machines in the LAN. The base station generates messages at a given rate. In each preview period, if there are any messages pending in its message queue, the base station sends a wake-up call to the client. The wake-up call is implemented using a tcp-connection. If the base station is unable to connect to the client during this period, the wake-up call is considered missing. This might happen due to other traffic in the LAN. In each edition, if wake-up call is sent, base station sends messages in its queue using a tcp-connection. Once a wake-up call is sent, messages generated within an edition do not have to wait till the next preview. The client waits for wake-up calls during each preview period. If there are any, it waits for messages during the following edition, otherwise it sleeps.

For the purpose of measuring the delay, birth-time stamps are attached to each message when it is generated. When the client receives the message it sends a reply to the base station which then measures the delay incurred in delivering the message. Overhead for shutting the receiver off and on is taken into account in measuring the fraction of time the receiver is off. For a given layout ratio and message rate, preview and edition are varied to measure the average delay and the activity ratio. Measurements are taken for High(1:10) and Low(1:100) layout ratios and High(120/hour) and Low(3/hr) message rates for messages of size 2Kbytes. Receiver shutoff overhead is the time it takes to configure the interface unit for reception when the unit is powered on. It is assumed to be of the order of a few milli-seconds.

[2]In Ethernet we can not guarantee the exact time of the message delivery, hence a very short preview time could result in a significant message loss

layout ratio 1:10 message rate 120/hr			
preview secs	edition secs	avg delay secs	activity ratio
0.5	5	6.080	0.258
1	10	5.760	0.360
5	50	18.654	0.483
10	100	34.937	0.545

layout ratio 1:100 message rate 120/hr			
preview secs	edition secs	avg delay secs	activity ratio
0.05	5	5.145	0.175
0.1	10	10.68	0.316
0.5	50	28.621	0.547
1	100	30.083	0.519

layout ratio 1:10 message rate 3/hr			
preview secs	edition secs	avg delay secs	activity ratio
0.1	1	1.107	0.092
1	10	15.950	0.099
5	50	18.215	0.133
10	100	35.666	0.176

layout ratio 1:100 message rate 3/hr			
preview secs	edition secs	avg delay secs	activity ratio
0.1	10	12.253	0.018
0.5	50	59.014	0.052
1	100	129.274	0.094

3.2.1 Interpretation of Results

Low message rates[3] result in receiver being off for upto 98% of the time. Hence, a message generated will have to wait till the next preview thus increasing average delay. Within the same layout ratio, for low preview sizes, the delay is comparable to size of edition since probability of missing the wake-up call is high. Lower layout ratios introduce larger delays because probability of no messages generated during preview is higher.

For high message rates, average delay did not grow proportional to the edition size.This is because once wake-up call is sent for an edition, messages generated during that edition do not have to wait till the next preview as the receiver is on in this edition. The activity ratio is high compared to low message rates. Layout ratios have lesser impact on the delay.

3.2.2 Conclusion

We find that implementing the level 3 of energy savings can, in practice, lead to quite dramatic energy savings. The Wavelan's receiver draws 1.8 W when on and 0.18 W when it is in doze mode (the card still has to be powered even if the receiver is off). The average e-mail and newsgroup user receives probably at most 30 to 40 messages a day and probably does not mind a few minute delay.

For instance, with the layout ratio $r = 0.01$, implemented with a preview size of 100 milli-seconds and an edition size of 10 seconds with message delay of 12 seconds, we could reduce the total energy consumption due to the activity of the receiver by an order of magnitude. For example, assuming that the mobile terminal is powered by 4 AA batteries each of which is capable of providing around 1 Watt for an hour we can keep the Wavelan receiver on for about 2.25 hours (and this ignores all the power drawn by other hardware elements!). If the ratio $r = 0.01$ leads to the activity of 2% then the same terminal can be receiving messages for about 19 hours.

We have ignored here the possible energy saving due to the CPU being in the doze mode. These savings are application dependent, since the CPU may have to run other applications which are not communication dependent. However, if the client is "communication intensive" then the savings due to the CPU being in the doze mode will have to be included as well.

3.3 Implementation of The Preview-Edition Protocol

In this subsection we discuss issues regarding which layers of the TCP/IP protocol stack should be made aware of the preview edition protocol. We take E-mail as an example application. It should be ensured that no changes are required in the existing network software on the Fixed Hosts in the Internet (except for the Mobile Support Stations).

Implementing the preview-edition protocol at the Application Layer means there are System Calls available : shutoff() and wakeup() for switching the receiver off and on at the Mobile Host. At the MSS, application level software is needed to buffer mes-

[3] 3 e-mail messages per hour constitute a low message volume from the hardware point of view, but actually a fairly intense e-mail use from the user's perspective.

sages (say, e-mail messages) until the next preview period, send destination IP Addresses during the preview and then send the messages during the edition period.

This obviously works only if there is an Indirect Protocol running at the Application layer. For example, for E-mail, currently a TCP connection is established between sender and receiver hosts. Now, there will have to be a RemoteHost-MSS-Mobile E-mail protocol. The MSS will receive E-mails on behalf of the Mobile Host and forward them using preview-edition.

On the other hand, this protocol can be implemented at the link layer. For this, any packets received by the link layer at MSS (say, Ethernet) will be buffered until the next preview period, their destination link-layer addresses will be sent during preview and the buffered packets are sent during edition. How this affects the throughput needs to be studied.

3.4 Applications of the Preview-Edition Protocol

Here we would like to discuss some of the possible applications of the preview-edition protocol. Our main motivation is, in fact, the electronic mail, but there are a number of related applications such as *talk* and *news*.

The main advantages of our protocol deal with the situations when communication is initiated by the MSS, not by the mobile host. Thus, applications like telnet and ftp, which are initiated by the client, are not going to benefit from our solution. In these cases we assume that the mobile host will stay active and will keep the receiver on during the (ftp, rlogin, telnet) session. On the other hand, in case of e-mail, we see dramatic savings. Instead of keeping the host's receiver on all the time, the receiver will be turned on and off periodically to listen to the preview report. If no messages are waiting at the MSS, then the receiver will be off for a large fraction of time (depending on the parameters of the protocol but roughly 90% or more). On the other hand, if the preview contains a *biff* message then the mobile host will have two basic choices: to request explicitly that the mail is downloaded from the MSS or to wait until the preview-edition protocol takes care of it without an explicit uplink request. The first solution corresponds to the mobile host executing "mail" command and the mail being downloaded by the MSS only upon receiving such a request from the host. We argue, in general, against such a solution: it will unnecessarily increase the uplink message traffic on the narrow uplink channel. In our solution, the MSS will anyhow do the best effort downloading of the message to the mobile host as soon as it can. In fact, the MSS should not even wake up the mobile host if it is not able to download the message in the next edition. In such a case the wake up message will be delayed until the MSS is capable of fulfilling the promise. Thus, there is no point in rushing the MSS by sending the mobile host initiated message, since the MSS cannot do better anyway. Notice that the user still has freedom to read his mail at any time. It is just that execution of the mail command will not result in the uplink message but rather in the local mail buffer access. The explicit uplink request is useful only in situations when the user wants to delay reading his messages until he is connected again. In such a case, waiting for the user's request can save wireless bandwidth. This assumes, however, that the message is not considered to be "urgent". We assume that the decision which messages are urgent and which are not has been made by the mobile user prior to "going wireless" and that the MSS is only considering urgent messages for wireless transmission anyhow. Yet another solution is that MSS include only message headers in the edition period and then allow the mobile user to explicitly request the selected subset of messages. In this case, the headers are received without the uplink request, while the messages are delivered to the mobile host responding to the explicit uplink request.

For applications like "talk" which are more real time, the client should probably switch on his receiver for the entire duration of the talk session. Otherwise, the energy saving is not enough to justify the increase in delay. Frequently switching the receiver on and off is an added overhead contributing more delay.

News applications (newsgroups, newsgroup updates) are more similar to e-mail and can be treated in a similar way. Perhaps the user can specify "urgent and important" news which have to be delivered as soon as they arrive and second priority news which can be wait to be seen after the client connects again.

3.5 Related Work : Differences and Similarities

As we said earlier, the preview-edition protocol has been motivated by the similar protocol used in cord-

less telephony. In satellite based paging solutions Motorola and Olivetti are considering keeping the pager's receiver "on" only at predefined periods of time which depend on the pager's identification. Therefore the activity periods are different for different classes of pagers. Also, there is no concept of edition and the solution is considered only for satellite based networks. In our case, we would like to implement it on variety of platforms, including wireless Ethernet and possibly, in the future, CDPD.

A similar protocol is mentioned in [7]. There, the mobile host keeps its receiver off until a message is to be received from the base station. At this time an uplink request is sent to the base station, indicating that the mobile unit is ready to receive data. This scheme would not work if the base station needs to send messages to the mobile unit asynchronously, i.e. without the mobile expecting them. A wake-up based protocol is required in such a case.

Our protocol has a "reservation protocol" flavor, but the concept of the reservation is very loose. Indeed, the reservation is only a "promise" of the MSS to deliver the packet (sometime) within the next edition. This promise could be broken. The only "reserved" slot is the preview period and this is the slot which is "polled" by the mobile clients.

Notice, the asymmetric nature of the preview-edition protocol which covers only the downlink channel. The uplink communication is regulated separately, since we are not concerned with the MSS power consumption. However, the preview-edition protocol reduces the uplink network traffic, if we avoid the explicit uplink requests by the mobile hosts (as recommended).

4 Conclusions

We have proposed two protocols which offer significant energy savings for applications such as newsgroups and electronic mail. The first protocol provides refined indexing of multcasted data and allows energy savings due to keeping the CPU in the doze mode. The preview-edition protocol allows, additionally, to keep the client's receiver off for most of the time. We have provided experimental results by implementing the above protocols on the application layer and testing them in a wireless LAN setting.

One can visualize a very simple combination of both protocols, when the client has to be first woken up before receiving the multicast group news. Further filtering within the multicast itself can be accomplished using the hashing based protocol described before.

We have implemented both protocols on the application layer, assuming that the features of switching the receiver off and forcing CPU into the doze mode are provided by the operating system.

References

[1] Ajay Bakre, B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", Technical Report DCS-TR-314, Department of Computer Science, Rutgers University, October 1994.

[2] T. F. Bowen et al., "The DATACYCLE Architecture," *Communications of the ACM*, Vol. 35, No. 12, pp. 71-81, December 1992.

[3] Stephen Deering, "Multicast Routing in a Datagram Internetwork," *phd thesis*, Stanford University, December 1991.

[4] R. Frenkiel "Private Communication"

[5] David Gifford et. al., "The application of digital broadcast communication to large scale information systems," *IEEE Journal on selected areas in communications*, Vol 3, No. 3, pp. 457-467, May 1985.

[6] David J. Goodman, "Trends in Cellular and Cordless Communications," *IEEE Communications Magazine*, pp. 31-40, June 1991.

[7] Andy Harter and Frazer Bennett, "Low Bandwidth Infra-Red Networks and Protocols for Mobile Communicating Devices", Technical Report tr.93.5, Olivetti Research Laboratory.

[8] T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, Vol. 37, No. 10, October 1994.

[9] T. Imielinski, S. Viswanathan and B. R. Badrinath, "Energy Efficient Indexing on Air," *Proc of ACM-SIGMOD, International Conference on Data Management*, Minnesota, pp. 25-36, May 1994.

[10] T. Imielinski, S. Viswanathan and B. R. Badrinath, "Power Efficient Filtering of Data on Air," *Proc of 4th International Conference on Extending Database Technology*, Cambridge-U.K, pp. 245-258, March 1994.

[11] B. Oki et al., "The Information Bus - An architecture for Extensible Distributed Systems," *Proc of ACM-SIGOPS*, December 1993.

[12] Samuel Sheng, Ananth Chandrasekaran, and R. W. Broderson, "A portable multimedia terminal for personal communications," *IEEE Communications Magazine*, pp. 64-75, December 1992.

[13] Lee, William C. Y., "Mobile Cellular Telecommunications Systems," *New York: McGraw-Hill*, 1989.

[14] J.W. Wong., "Broadcast Delivery," *Proceedings of the IEEE*, Vol. 76, No. 12, pp. 1566-1577, December 1988.