UGA–CS–TR–01–001

Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats

Surendar Chandra

Department of Computer Science
University of Georgia
Athens, GA 30602-7404

October 2001

Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats

Surendar Chandra

Department of Computer Science
University of Georgia
Athens, GA 30602-7404
surendar@cs.uga.edu

Abstract

The proliferation of multimedia-capable mobile devices and ubiquitous high-speed network technologies to deliver multimedia objects is fueling the demand for mobile streaming multimedia. However, the end-user experience is severely constrained by the available battery capacity on the mobile clients. Wireless network interface cards (WNIC) consume significant amounts of power. Since streaming applications tend to be long running, WNIC energy consumption is particularly an acute problem. In this work, we explore the WNIC energy consumption implications of popular multimedia streaming formats from Microsoft (Windows media), Real (Real media) and Apple (Quick Time). We investigate the energy consumption under varying stream bandwidth and network loss rates. We also explore history-based client-side strategies to reduce the energy consumed by transitioning the WNICs to a lower power consuming *sleep* state.

We show that Microsoft media tends to transmit packets at regular intervals. For high bandwidth streams, Microsoft media exploits network-level packet fragmentation, which can lead to excessive packet loss (and wasted energy) in a lossy network. However, the regularity of packet arrival rates facilitates history-based client-side policies to transition to lower power states. We show that a Microsoft media stream optimized for 28.8 Kbps can save over 80% in energy consumption with 2% data loss. A high bandwidth stream (768 Kbps) can still save 57% in energy consumption with less than 0.3% data loss. Real stream packets tend to be sent closer to each other, especially at higher bandwidths. Quicktime packets sometimes arrive in quick succession; most likely an application level fragmentation mechanism. Such packets are harder to predict at the network level without understanding the packet semantics. Our work enables multimedia proxy and server developers to suitably customize the stream to lower client energy consumption.

1 Introduction

The proliferation of inexpensive, multimedia capable mobile devices and ubiquitous high-speed network technologies to deliver multimedia objects is fueling the demand for mobile streaming multimedia. Public venues [31] and sports stadiums [29] are deploying high speed IEEE 802.11b [20] based public wireless LAN networks. Commodity PDA devices that allow the users to consume mobile streaming multimedia are becoming popular. For example, the iPAQ [1] offers a 206 MHz Intel StrongARM processor, 240x320 color display, and 64 MB of RAM storage; enough compute resources to render high quality video streams. The iPAQ utilizes conventional PCMCIA cards to access a 802.11b WLAN network as well as 3G wireless networks. Versions of media players from Microsoft [24] and PacketVideo [26] allow iPAQ users to consume streaming media in Microsoft Media and MPEG-4 formats, respectively.

Such advances in hardware and software technologies have not been matched by corresponding improvements in battery technologies. A necessary feature for mass acceptance of a streaming multimedia device is acceptable battery life. Typical mobile devices have utilized conventional primary batteries as well as re-chargeable Lithium polymer and metal hydride batteries. The usefulness of these devices in watching streaming multimedia is restricted by the battery capacity. Future trends in battery technology do not promise dramatic improvements that will make this issue disappear.

Earlier work by Stemm et al. [32] report that the network interfaces draw significant amounts of power. For example, a 2.4 GHz Wavelan card consumes 177 mW while in *sleep* state, but consumes 1319 mW while *idle*. Havinga et al. [14] note that this Wavelan card consumes 1425 mW for receiving data and 1675 mW for transmitting data. A transition from *sleep* to *idle* takes 250 μs (with power consumption of the *idle* state). Streaming media tends to be large and long running and consume significant amounts of battery resources to download data. Hence, it is important to look at techniques to reduce the energy consumed by the network interface to download the stream.

Traditionally, reducing the fidelity of the stream and hence the size is a popular technique that is used to customize the multimedia stream for a low bandwidth network. Reducing multimedia fidelity can also be expected to reduce the amount of data and hence the total energy consumed. However, if care is not taken to return the network interface to the *sleep* state as much as possible, reducing the amount of transmitted data will have negligible effect on the overall client energy consumption. Frequent switching to low power consumption states also promises the added benefit of allowing the batteries to recover, exploiting the battery recovery effect [7].

In this work, we assume that content providers broadcast multiple versions of the same stream with varying fidelities. The users choose a particular stream based on their available battery capacity and projected future usage requirements. Users can manually choose the appropriate stream or the infra-structure can automatically redirect the user to the correct stream. The specific mechanisms on how the user chooses a particular stream is beyond the scope of this work. Also, the IEEE 802.11b standard [20] allows for reducing the transmit power by up to four power levels. In this work, we primarily focus our attention on client side behavior. Since the clients mostly receive data, we do not explore reducing the transmitter power as a viable technique to reduce client side energy consumption.

Our goal is to understand the energy consumption aspects of changing the fidelity of popular multimedia streaming mechanisms. We explore the energy implications of these streaming formats under varying network conditions. Based on our observations, we develop history based client-side techniques to exploit the stream behavior and lower the energy required to receive these streams. For our experiments, we explore Microsoft media [24], Real media [27] and Quicktime [3] as popular streaming formats. We configured the media services to utilize UDP packet streams for data transport. We believe that these widely popular formats are more likely to be deployed than custom streaming formats (that are specially optimized for lower energy consumption).

We show that Microsoft media tends to transmit packets at regular intervals. For high bandwidth streams, Microsoft media exploits network level fragmentation, which can lead to excessive packet loss (and wasted energy) in a lossy network. However, the regularity of packet arrival rates allow simple, history based client side policies to transition to lower power states with minimal data loss. We show that a Microsoft media stream optimized for 28.8 Kbps can save over 80% in energy consumption with 2% data loss. A high bandwidth stream (768 Kbps) can still save 57% in energy consumption with less than 0.3% data loss. Real stream packets tend to be sent closer to each other, especially at higher bandwidths. Quicktime packets sometimes arrive in quick succession; most likely an application level fragmentation mechanism. Such packets are harder to predict at the network level without understanding the semantics of the packets themselves. We believe that modifying Real and Quicktime services to transmit larger data packets at regular intervals can offer better energy consumption characteristics with minimal latency and jitter. Our work enables multimedia proxy and server developers to suitably customize the stream to lower client energy consumption.

The remainder of this paper is organized as follows: we present the experimental setup, evaluation methodologies, measurement metrics and the workloads used in our study in Section 2. Section 3 analyzes the stream behavior and energy consumption characteristics under different network conditions. Section 4 explores client side policies to reduce the energy required to receive these multimedia streams. Section 5 describes related work. We present our conclusions and outline future research directions in Section 6.

2 Experiment Objectives and Design

2.1 Objectives

We designed our experiments to answer the following questions:

- What are the client energy consumption implications of viewing multimedia in popular streaming formats?
- Can we develop client-side strategies to reduce energy consumption? In general, client-side policies require minimal modifications to the streaming server.

2.2 Multimedia Stream Collection

For our experiments, we utilized the Wall (movie) theatrical trailer. We replayed the trailer from a DVD player and captured the stream using the Dazzle Hollywood DV Bridge. We utilized Adobe Premiere 6.0 to convert the captured DV stream to the various streaming formats. The trailer was 1:59 minutes long.

The Wall trailer can be digitized to a high quality stream and hence allowed us the flexibility of creating streams of varying fidelities. Hence, we utilize this stream for the rest of this paper. We experimentally confirmed that our results are applicable for other publically available streams (of lower quality settings).

2.3 Experiment setup

The system setup that was used to analyze the energy consumption characteristics of popular streaming formats is illustrated in Figure 1. The various components of our system are:

- **Multimedia Server**: Our multimedia server (Dell 330) was equipped with a 1.5 GHz Pentium 4 with 512 MB of PC800 RDRAM memory, running Microsoft Windows 2000 Server SP2. The server was running Windows Media Service, Realserver 8.0 and Apple Darwin Server (preview version 3.0).

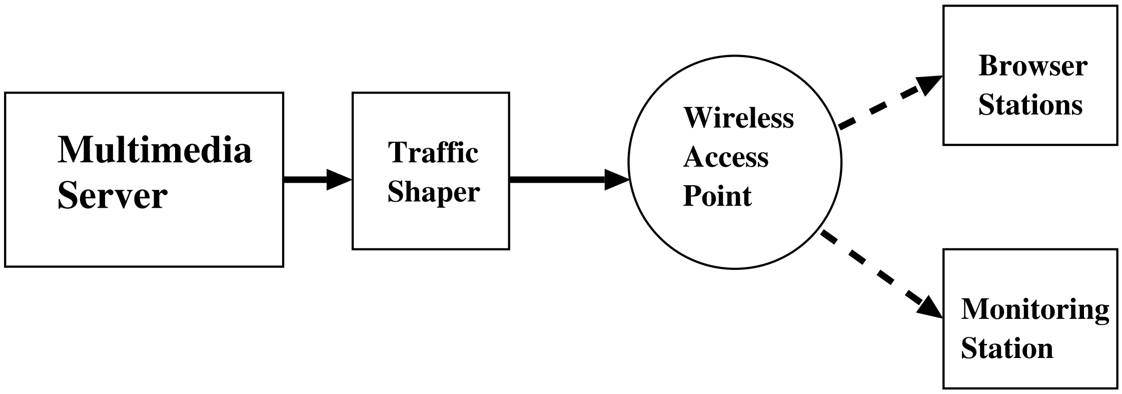


Figure 1: Experiment Setup

- **Traffic Shaper**: The various network conditions were simulated using a dummynet based FreeBSD traffic shaper. The traffic shaper host (Dell 4400) consisted of a dual 933 MHz Pentium III Xeon with 1.5 GB of main memory, running FreeBSD 4.3 (STABLE).
- **Wireless Access Point**: For our experiments, we utilized a dedicated Orinoco AP500 access point (802.11b) with an external range extender antenna. Throughout our experiments, we had turned off the security encryption feature of the access point.
- **Browser Stations**: For our experiments, we utilized an IBM T21 laptop with 800 MHz Pentium III processor, 256 MB RAM and running Windows 2000 Pro SP2. Wireless connectivity was provided by an 11 Mbps Orinoco PCMCIA WLAN card. The laptop accessed the streaming formats using Microsoft media, Real and Quicktime players.

  We also experimented with a Compaq iPAQ 3650 Pocket PC with 32 MB RAM running Windows CE 3.0 SP1. Wireless networking was provided using an external (powered separately) PCMCIA sleeve and a 11 Mbps Orinoco PCMCIA card. The iPAQ accessed Microsoft media using Media Player 7. We noticed that the media stream for the iPAQ player was similar to the laptop player. In the interest of space, we only discuss results from the laptop player.
- **Monitoring Station**: The packets transmitted from the server to the browser station was passively captured by the monitoring station, which was physically kept close to the browser station and the wireless access point. The monitoring station was a 500 MHz Pentium III laptop with 256 MB RAM and running Redhat Linux 7.0.

  Packets were capturing using tcpdump 3.6. It should be noted that the time stamps captured by tcpdump are the application level packet delivery times (not the time that the packet reached the network interface card). This is acceptable for our purposes because, any client side adaptation systems will also receive the packets with the same time stamps as tcpdump (and not the time that the packet reached the network interface).

2.4 Client-side History-based Policy

While developing the client-side adaptation policies, we envisioned a client-side proxy architecture that allows the flexibility of manipulating the network interface power states without actually modifying the media browsers or the servers themselves. Throughout our experiments, we utilized the following published power parameters ([14, 32]) for a Wavelan 2.4 GHz WNIC card: 177 mW - *sleep* state, 1319 mW - *idle* state,

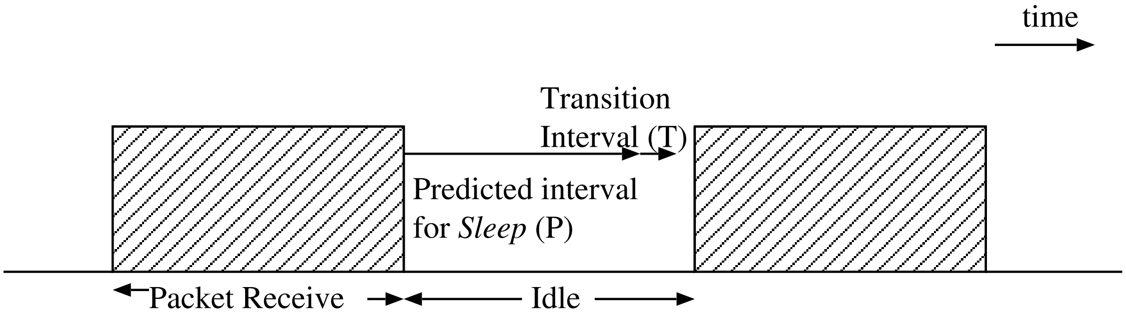


Figure 2: Stream network packet states (simplified)

1425 mW - receiving state and 1675 mW - transmit state. We assumed that the wireless network provides a useful throughput of 4 Mbps. We also assumed that a transition from sleep to idle took 250 μsec.

The various states of packet reception is shown by a simple illustration in Figure 2. Traditionally, the network interface switches between Packet *receive* state and *idle* state. We explore techniques that can transition the interface to the *sleep* state during the *idle* state. If the predicted *sleep* interval was too conservative, then the network interface spends the rest of the time in *idle* state, potentially missing on further energy saving. If the predicted *sleep* interval was too aggressive, then the network interface might be asleep while a packet was being delivered, potentially missing a data packet.

History based policies predict the required *sleep* interval by averaging the *idle* times over the past *History receive-idle* cycles. We vary the dependence on past history by offsetting the average with a small *threshold* such that $Predicted\ sleep\ interval = \frac{\sum_{History} Past\ idle\ times}{History} - threshold$. While receiving fragmented network packets, the idle times are only computed for the first fragment. The network interface waits for all subsequent fragments without a transition to the *sleep* state.

2.4.1 Performance metrics

For our experiments, we utilize the following performance metrics to measure the efficacy of our approach:

- **Energy consumed:** The goal of these experiments is to reduce the energy consumed by the WNIC.
- **Percentage of data bytes dropped:** This metric gives an indication of the mis-predictions. The goal is to keep dropped data packets to a minimum.

3 Energy Consumption Characteristics of Popular Streaming Formats

In this section, we perform a detailed analysis of the energy implications of receiving network packets for popular streaming formats. In the next section, we explore schemes that can be employed by a client-side proxy to conserve client WNIC energy consumption.

We performed our experiments by streaming variations of the Wall trailer that was transcoded for various network bandwidth requirements. The cumulative multimedia data transferred to the client and the feedback transmitted from the client to the server for Microsoft Media, Real and Quicktime are tabulated in Tables 1(a), 1(b) and 1(c), respectively. Overall, we note that the three formats send comparable amounts of multimedia data to the client. However, Quicktime utilizes more feedback bandwidth compared to Real or Microsoft media.

Stream bandwidth	**Transmit** (MB)	**Receive** (MB)
28.8 Kbps	0.0041	0.2657
56 Kbps	0.0071	0.4043
128 Kbps	0.0039	1.4851
256 Kbps	0.0041	3.2936
768 Kbps	0.0041	8.6783
2 Mbps	0.0042	26.3031

(a) Microsoft Media (laptop)

Stream bandwidth	**Transmit** (MB)	**Receive** (MB)
28 Kbps	0.0108	0.3428
56 Kbps	0.0102	0.5694
128 Kbps	0.0099	1.2894
256 Kbps	0.0100	3.504
512 Kbps	0.0109	6.8335

(b) Real (laptop)

Stream bandwidth	**Transmit** (MB)	**Receive** (MB)
28.8 Kbps	0.06499	0.4718
56 Kbps	0.0604	0.6207
128 Kbps	0.0630	1.2442
256 Kbps	0.07590	3.6500

(c) Quicktime (laptop)

Table 1: Amount of data transmitted

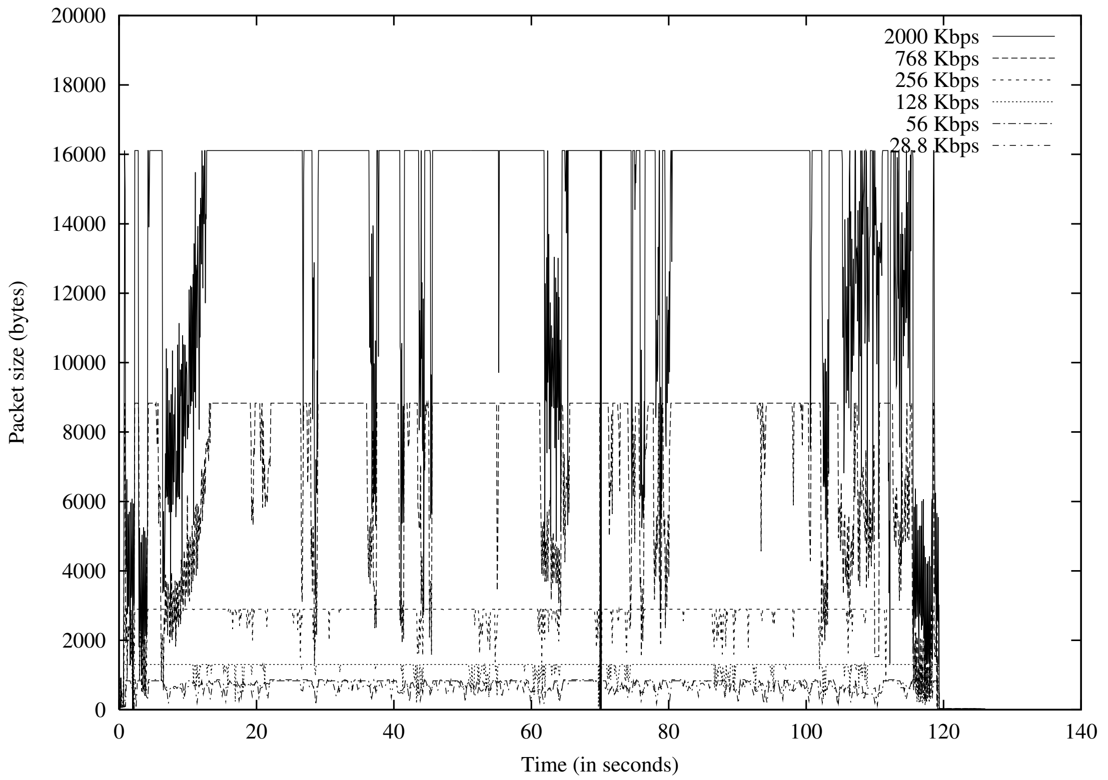


Figure 3: Microsoft Media - Packet lengths to client on a network that does not drop any packets

3.1 Microsoft Media

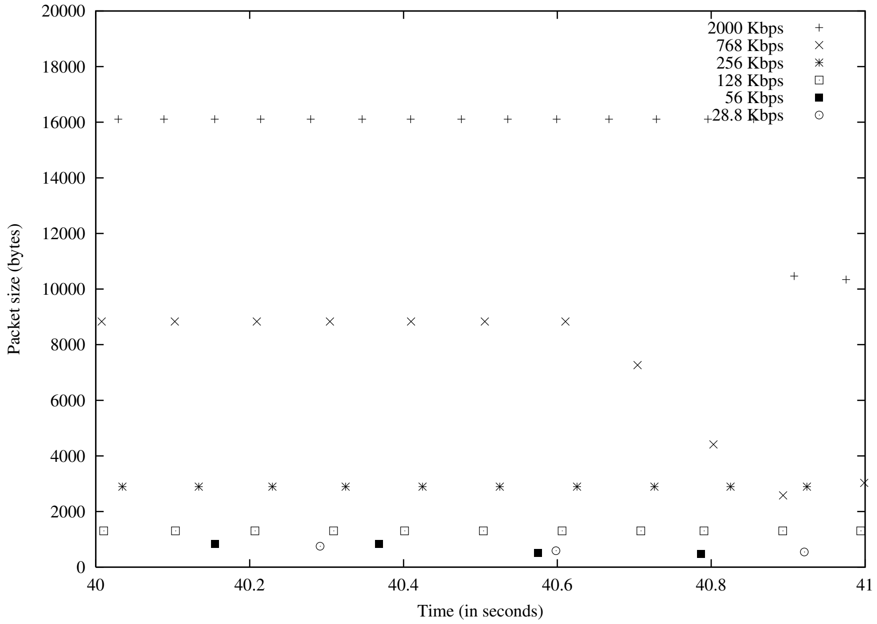
In this section, we discuss our results from streaming Microsoft media. First we present the results of tracing the packets for the Wall video clip under a network with no packet loss. Next we present the results for a network which drops 5% of the packets. We also present the energy consumed by the WNIC in receiving these streams. In the following sections, we discuss similar results for transmitting the multimedia stream in Real and Quicktime formats.

3.1.1 Network with no packet loss

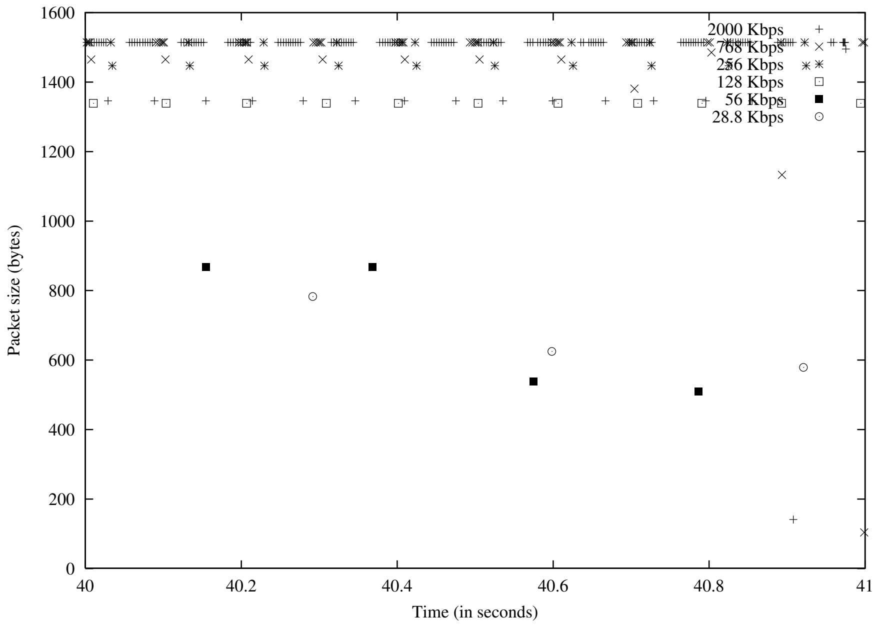
For our experiments, the laptop player requested unicast Microsoft media streams (MMS - UDP) that were optimized for various available network bandwidths. The sizes of the packets received by the laptop with time are plotted in Figure 3. We note that, for the first 10 seconds or so, the streams progressively adapted to consume the slated overall network bandwidth.

We note that once the streams stabilize, they request packets of similar lengths; higher quality streams result in larger packet sizes. Streams optimized for 28.8 Kbps, 56 Kbps, 128 Kbps, 256 Kbps, 768 Kbps and 2 Mbps request packets of sizes 862 bytes, 834 bytes, 1305 bytes, 2893 bytes, 8831 bytes and 16112 bytes, respectively. The underlying network layers fragment such large packets to smaller fragments. In order to better visualize the stream dynamics, we highlight the packet lengths for the duration of 40 through 41 seconds since the start of the stream as well as the network packets (after fragmentation) in Figures 4(a) and 4(b) respectively.

From Figure 4(a), we see that the media server tends to transmit packets at seemingly constant intervals. Such constant packet intervals can assist us in developing client-side policies to transition the WNIC to *sleep*



(a) Packet lengths (closeup - laptop)



(b) Network packet lengths (closeup - laptop)

Figure 4: Microsoft Media - Closeup view of packets to client from 40:00 through 41:00 seconds since start of stream

Stream bandwidth	Energy consumed (in Joules)	
	Network - 0% loss	Network - 5% loss
28.8 Kbps	157.497	157.687
56 Kbps	157.562	157.634
128 Kbps	157.709	159.107
256 Kbps	160.237	159.232
768 Kbps	163.229	164.194
2 Mbps	174.768	163.923

Table 2: Energy consumed by the WNIC for receiving Microsoft media streams (in Joules)

state often. From Figure 4(b), we see that the underlying network fragments the packets (packets traversed an Ethernet network with a MTU of 1500 bytes). These fragments are transmitted back-to-back in quick succession. Such dependence on network level fragmentation of UDP packets restricts the resiliency of Microsoft media services in a noisy channel.

3.1.2 Network with 5% packet loss

Next, we utilized the dummynet interface in the traffic shaper node to randomly drop 5% of the network packets. The packet lengths of multimedia streams customized for various client network bandwidths on a network that randomly drops 5% of the packets are shown in Figure 5(a). A closeup view of the network packets for duration between 40 and 41 seconds since the start of the stream are shown in Figure 5(b). High bandwidth Microsoft media streams are adversely affected by dropped fragments. The stream oscillates between various client bandwidths, frequently changing the client stream quality. Frequently a 2000 Kbps stream adapts to a 131 Kbps stream. Such frequent changes can adversely affect any adaptation policies explored in Section 4. Such fragmentation also increases the wasted energy consumption at the client, the network protocol stack would drop 10 good frames (of size 1500 bytes each) just because the 11th frame (as part of a 16112 byte packet) was lost in transmission.

3.1.3 Energy consumed by the WNIC interface

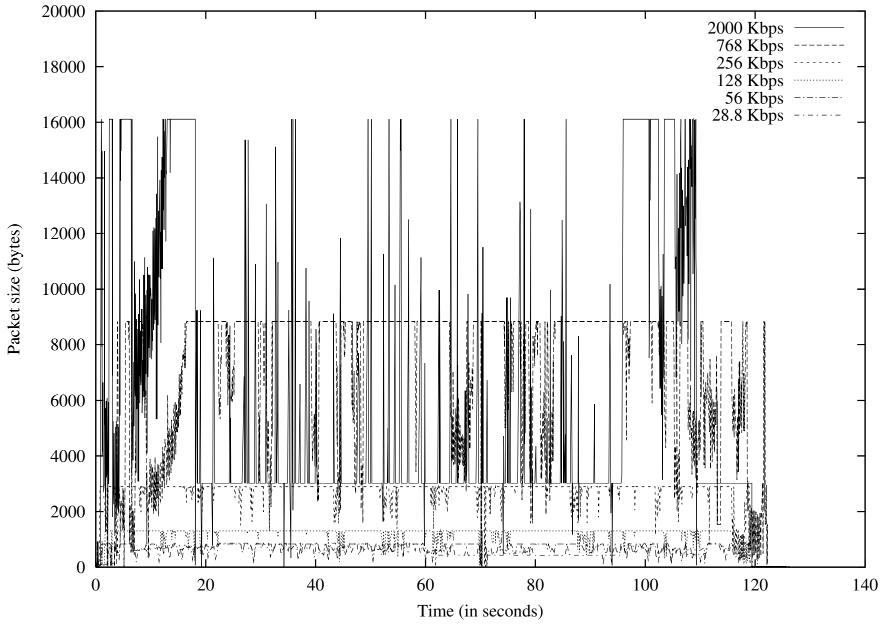
The cumulative energy consumed for the various Microsoft media streams are tabulated in Table 2. Our network card consumes almost the same power to *receive* as well as in *idle* state. Hence, as expected, there was a modest difference in the overall energy consumption among the various transcoding levels, even though there was two orders of magnitude difference in the amount of data transmitted (from top to the bottom). This effect was also noted by Stemm et al. [32]

3.2 Real

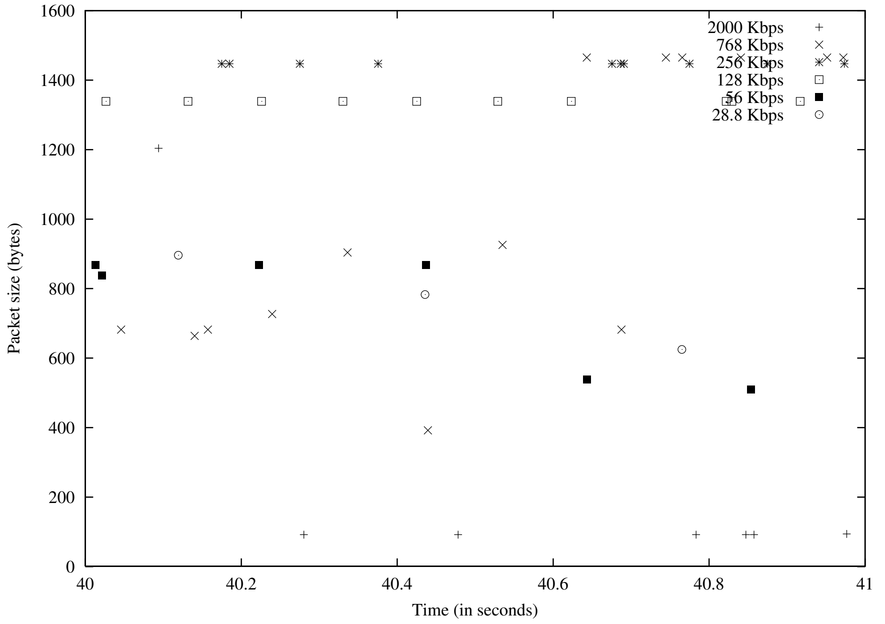
In the last section (Section 3.1), we analyzed the behavior of Microsoft media streams. In this section, we perform similar analysis for Real media streams under varying network conditions and bandwidth requirements. We will continue with similar analysis for Quicktime in the next section (Section 3.3).

3.2.1 Network with no packet loss

For our experiments, the laptop real player requested UDP streams that were optimized for various network bandwidths. The player was configured to turn off all local buffering. Figure 6(a) plots the size of network packets received by the laptop with time. A closer look at the size of packets received in the interval 40

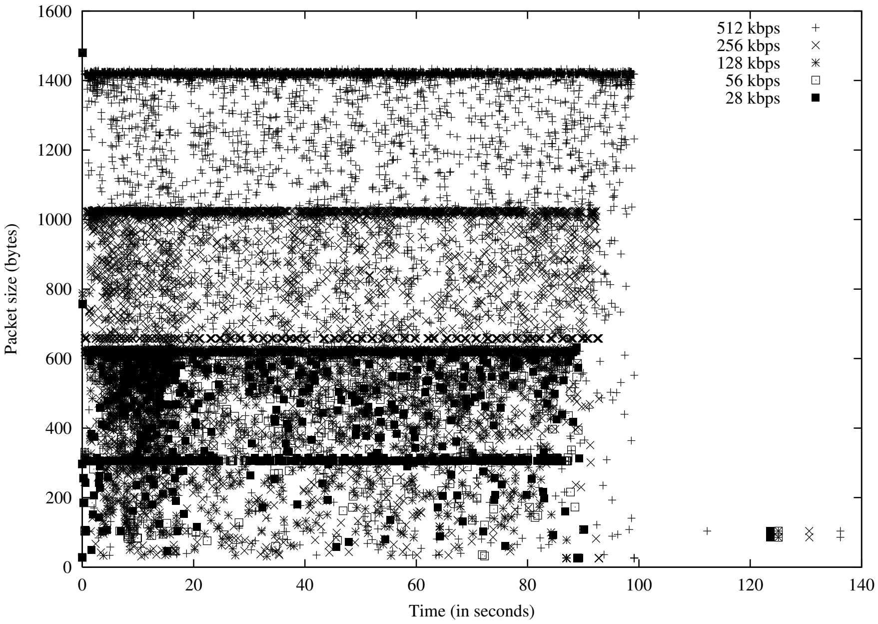


(a) Packet lengths (laptop)

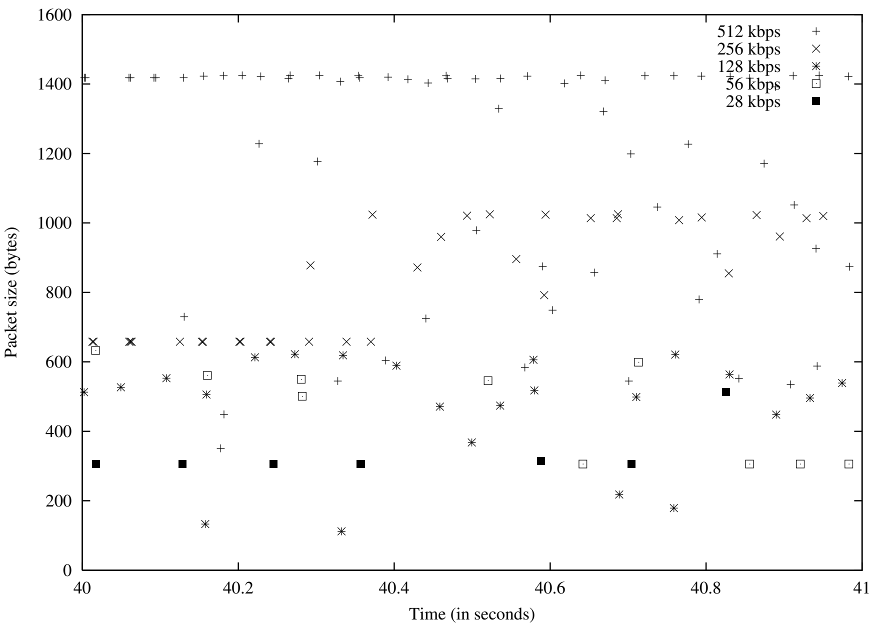


(b) Network packet lengths (closeup - laptop)

Figure 5: Microsoft Media - Client on network with 5% random packet loss



(a) Packet lengths (laptop)



(b) Packet lengths (closeup - laptop) (closeup from 40 through 41 sec. since start)

Figure 6: Real - Packet lengths to client on a network with 0% packet loss

Stream bandwidth	Energy consumed (in Joules)	
	Network - 0% loss	Network - 5% loss
28.8 Kbps	119.24	136.074
56 Kbps	119.151	136.226
128 Kbps	119.203	174.654
256 Kbps	124.308	156.293
512 Kbps	133.231	160.202

Table 3: Energy consumed by the WNIC for receiving Real streams (in Joules)

through 41 seconds since the start of the stream is illustrated in Figure 6(b). From Figure 6(a), we notice that the packet sizes are more variable than the packet sizes for Microsoft media (Figure 3). Individual packets are well within 1500 bytes (MTU of the Ethernet segment) and hence are not fragmented by the network. The packets tended to be transmitted at much closer intervals than for the Microsoft media packets. However, at the network level, the packets tend to be evenly distributed (as compared to Microsoft media) because of the lack of network fragmentation. It is interesting to note that most of the packets were received within 100 seconds and the player essentially buffered the last 20 seconds of the data (even though the player was configured to disable any local buffering).

3.2.2 Network with 5% packet loss

Next, we utilized our dummynet infrastructure to randomly drop 5% of the network packets and repeated the experiment from Section 3.2.1. The resulting packet lengths are plotted in Figure 7(a). A closer look at the packets from time 40 through 41 seconds since the start are plotted in Figure 7(b). From Figure 7(a) we note that high quality streams show less variability than when there was no packet loss. The amount of data received is also reduced as the stream adapts to the lossy network. From Figure 7(b), we note that the high quality stream packets are transmitted less often than for a loss-less network, indicating an adaptation to a lower quality stream.

3.2.3 Energy consumed by the WNIC interface

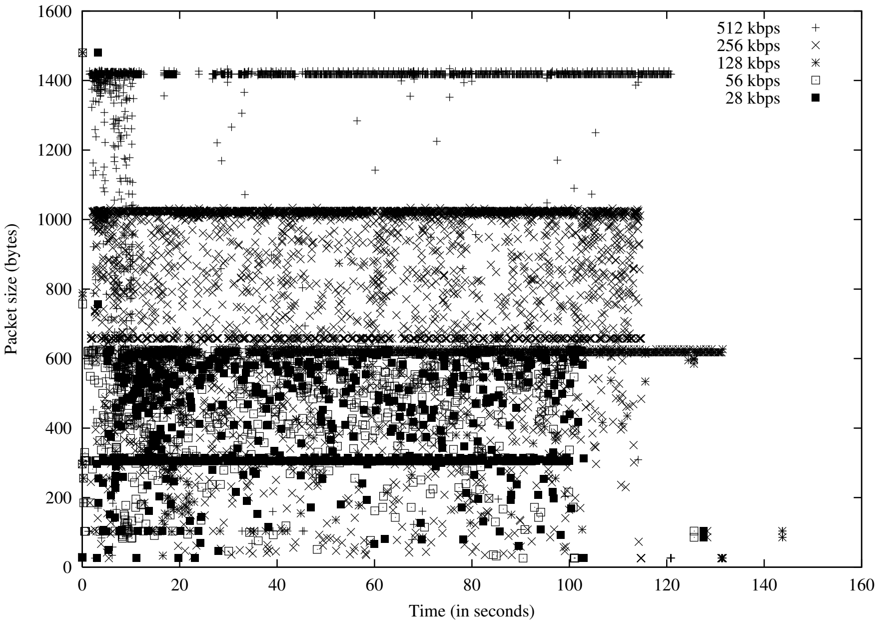
The cumulative energy consumed for the various Real streams are tabulated in Table 3. As discussed earlier (Section 3.2.1), the entire Real stream was transmitted in less than 100 seconds (instead of 119 seconds). Hence, Real avoids extra *idle* states and (apparently) consumes less energy than Microsoft Media (Table 2). As discussed earlier (Section 3.1.3), for a lossless network, reducing the stream bandwidth leads to modest reductions in energy required by the network interface (for an order of magnitude reduction in the amount of data). On a lossy network, the energy consumption fluctuates as the network tries to adapt to the lossy nature of the network.

3.3 Quicktime

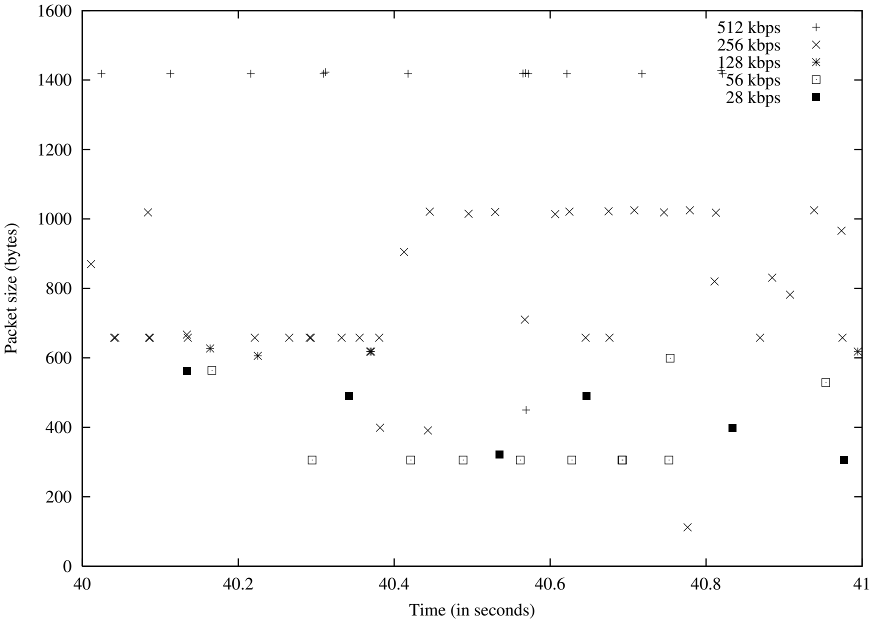
In the last two sections (Sections 3.1 and 3.2), we analyzed the behavior of Microsoft media and Real streams. In this section, we continue our investigation into Quicktime streams.

3.3.1 Network with no packet loss

The laptop Quicktime player requested the UDP streams at various transcoding levels. Throughout the experiments, local buffering in the player was disabled. Figure 8(a) plots the size of network packets received

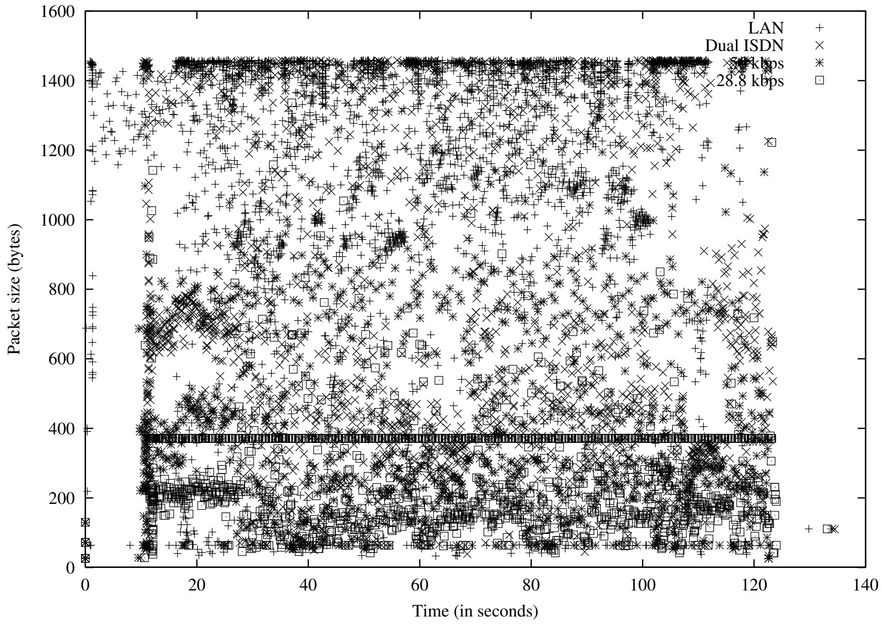


(a) Packet lengths (laptop)

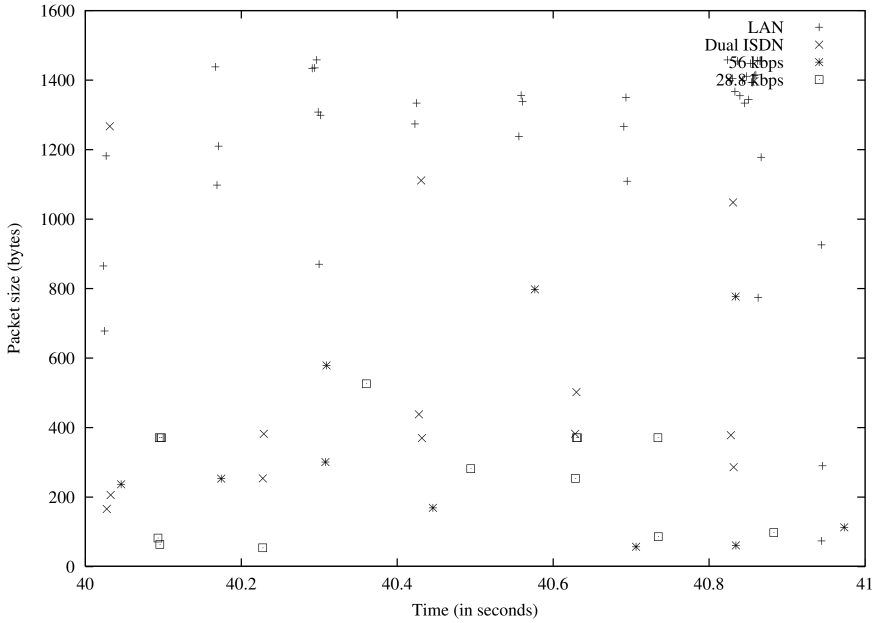


(b) Packet lengths (closeup - laptop)

Figure 7: Real - Packet lengths to client on network with 5% packet loss



(a) Packet lengths (laptop)



(b) Packet lengths (closeup - laptop)

Figure 8: Quicktime - Packet lengths to client on a network that does not drop any packets

Stream	**Energy consumed (in Joules)**	
bandwidth	**Network - 0% loss**	**Network - 5% loss**
28.8 Kbps	149.657	149.974
56 Kbps	149.735	150.006
128 Kbps	150.081	149.859
256 Kbps	149.239	152.877

Table 4: Energy consumed by the WNIC for receiving Quicktime streams (in Joules)

by the laptop with time. A closer look at the size of packets received in the interval 40 through 41 seconds since the start of the stream is illustrated in Figure 8(b). From Figure 8(a), we notice that the packet sizes are even more variable than the packet sizes for Microsoft media and Real (Figures 3 and 6(a), respectively). Individual packets are well within 1500 bytes (MTU of the Ethernet segment) and hence not fragmented by the network. High bandwidth streams tend to consist of packets that were sent in quick succession followed by extended idle intervals (probably an application level fragmentation mechanism). In general, such packet behavior are harder to predict at the network level without understanding the packet semantics.

3.3.2 Network with 5% packet loss

Next, we utilized our dummynet infrastructure to randomly drop 5% of the network packets and repeated the experiment from Section 3.3.1. The resulting packet lengths are plotted in Figure 9(a). A closer look at the packets from time 40 through 41 seconds since the start are plotted in Figure 9(b). From Figure 9(a) we note similar effects of application level fragmentation mechanisms (discussed in Section 3.3.1).

3.3.3 Energy consumed by the WNIC interface

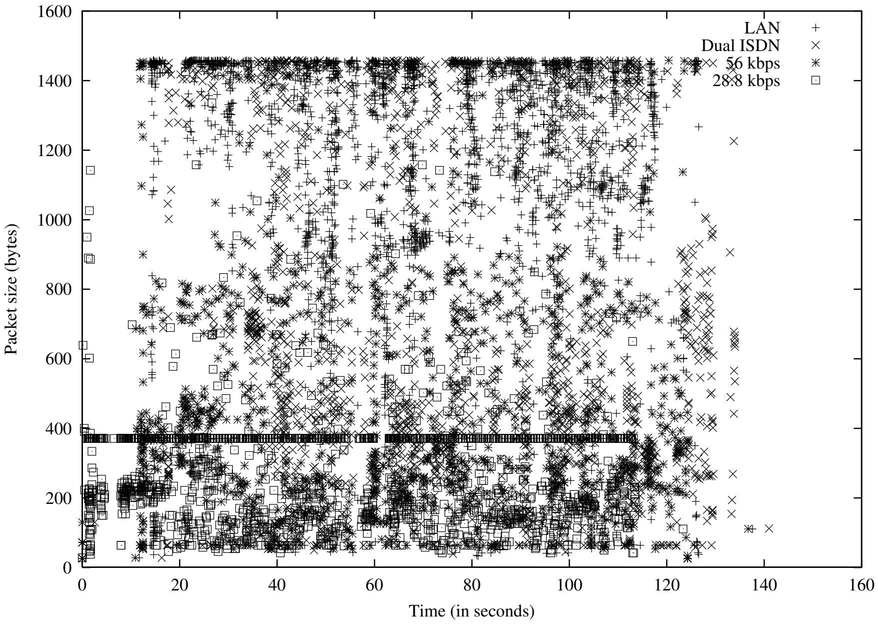
The cumulative energy consumed for the various Quicktime streams are tabulated in Table 4. As discussed earlier (Section 3.1.3), there was little difference in the overall energy consumption among the various streaming transcoding levels (even though there was an orders of magnitude difference in the amounts of data received).

To summarize, in this section we analyzed the energy implications of receiving network packets for the various streaming formats. We noted that Microsoft media transmitted large packets at fairly regular intervals. Some of these large packets are fragmented in the network layers. Such fragmented packets tend to be transmitted together; affecting the inter-packet arrival times. Real and Quicktime tend to show more variation in the packet sizes and packet inter-arrival times. Real transmits most of the data in 100 seconds and hence (apparently) consumes less energy to receive the stream. Quicktime packets were transmitted in quick succession followed by prolonged idle times, signifying an application level packet fragmentation mechanism. Such packet behavior are harder to predict at the network level without understanding the packet semantics.

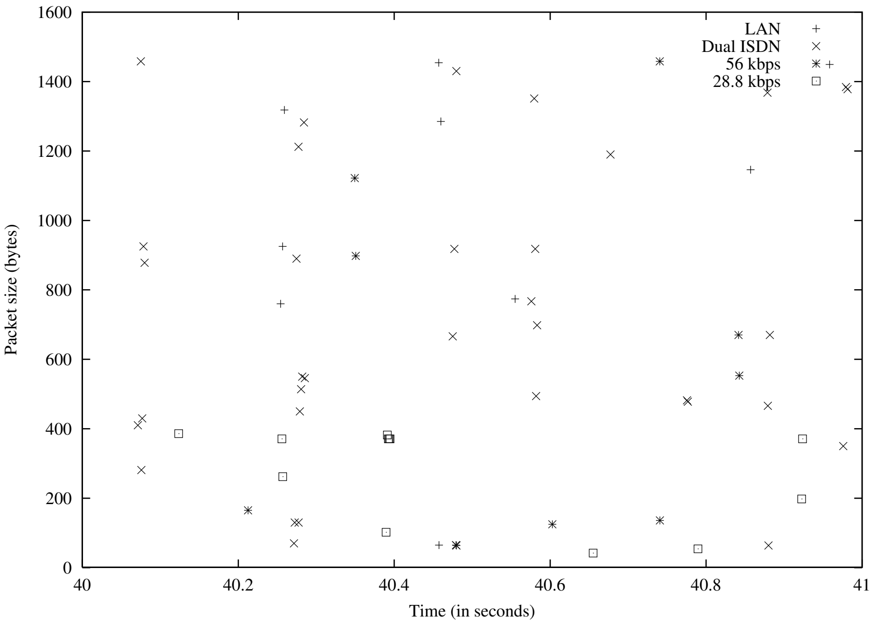
4 Strategies for adapting client network interface to lower power state

In the previous section, we analyzed the packet reception behavior for the various streaming formats under changing network conditions. In this section, we exploit some of these transmission characteristics to develop policies to aggressively transition the network interface into a lower power *sleep* state.

In a 802.11b wireless network, if a network interface is in *sleep* mode during a packet transmission, the packet is not received at the client. So it is imperative that the network interface be ready for the packet.



(a) Packet lengths (laptop)



(b) Packet lengths (closeup - laptop)

Figure 9: Quicktime - Packet lengths to client on network with 5% packet loss

Stream Format	Stream bandwidth	Network (0% Packet loss)		Network (5% Packet loss)	
		Idle	Receive	Idle	Receive
Microsoft Media	28.8 Kbps	122.083	0.552644	122.718	0.491834
	56 Kbps	118.73	0.834876	122.232	0.840554
	128 Kbps	116.384	3.04786	117.486	3.0575
	256 Kbps	116.276	6.75366	115.194	7.1352
	768 Kbps	104.732	17.7815	97.1372	21.8771
	2000 Kbps	69.6157	53.8773	102.945	19.7851
Quicktime	28.8 Kbps	138.085	1.10031	127.597	1.14442
	56 Kbps	132.297	1.39575	134.795	1.44227
	128 Kbps	135.917	2.67708	138.465	2.50058
	256 Kbps	125.669	6.76324	148.896	5.47502
Real	28 Kbps	89.6865	0.721112	102.435	0.732616
	56 Kbps	89.1371	1.18549	102.001	1.19389
	128 Kbps	87.6045	2.65937	131.396	1.04452
	256 Kbps	86.6539	7.19525	110.891	7.20237
	512 Kbps	109.968	14.0139	116.718	4.52506

Table 5: Idle times for the various streaming formats

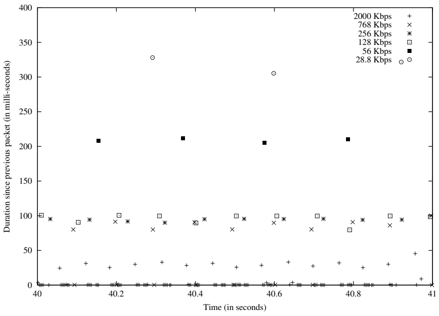
802.11b defines a power management mechanism wherein the network access point can buffer packets on behalf of the *sleeping* network interface. The client periodically wakes up on a beacon to receive the buffered packets. Such buffering increases the stream jitter and so we do not explore it in this paper. Our goal is to develop policies that allow the network interface to enter the *sleep* state as much as possible; without losing network packets.

In order to develop such transition strategies, first we tabulate the overall idle slots in our stream traces in Table 5. We note that for low quality streams, the network interface is idle for almost all the time. Even for high quality streams, the network interface is idle for more than half the time. Thus, it should be possible to save at least 50% of *idle* power.

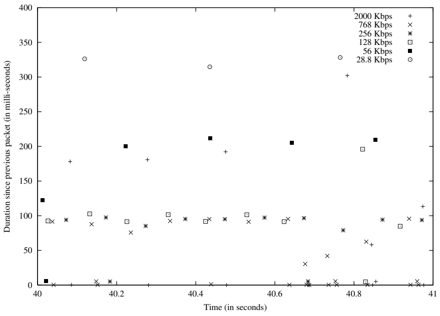
We also plot the inter-packet arrival times for the various streaming formats under changing network conditions in Figure 10. In Figure 10(a), we note the presence of fragmented network packets for Microsoft media streams (consecutive packets of 0 ms inter-arrival times). In general, the inter-packet arrival times for Microsoft media streams are fairly constant. Quicktime tends to send multiple packets right after each other, with inter-packet arrival times of 0 ms (Figure 10(e). High quality Real stream sends packets with small inter-packet arrival times (less than 50 msec) (Figure 10(c)). Predicting inter-arrival times offer us an opportunity to transition the interface to the *sleep* state to conserve energy. We will present our results for the client-side history-based policy outlined in Section 2.4.

4.1 History based policies

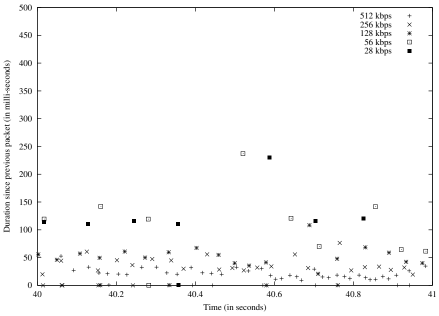
Recall that history based policies predict the required *sleep* interval by averaging the *idle* times for the past *History receive-idle* cycles. We vary the dependence on past history by offsetting the average with a small *threshold* such that $Predicted\ sleep\ interval = \frac{\sum_{History} Past\ idle\ times}{History} - threshold.$



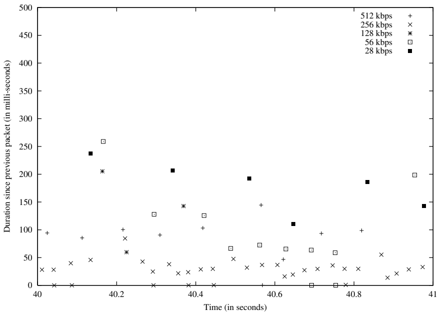
(a) Microsoft media - 0% packet loss



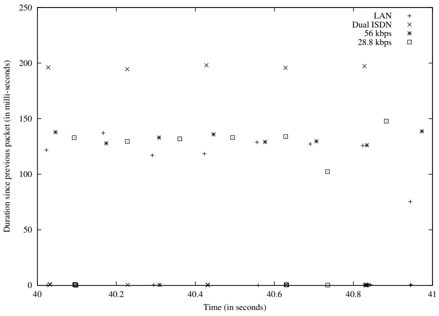
(b) Microsoft media - 5% packet loss



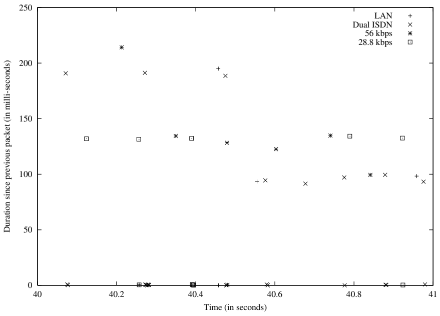
(c) Real - 0% packet loss



(d) Real - 5% packet loss

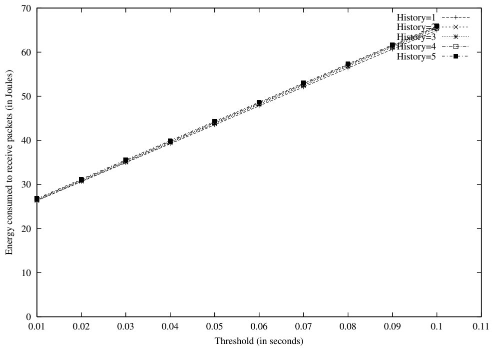


(e) Quicktime - 0% packet loss

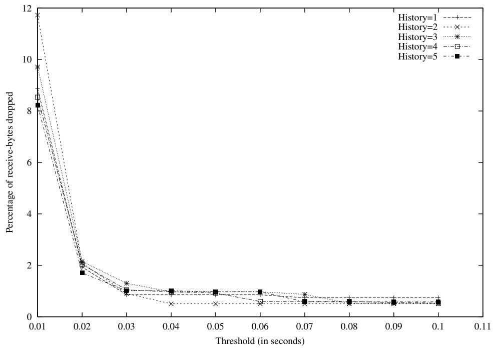


(f) Quicktime - 5% packet loss

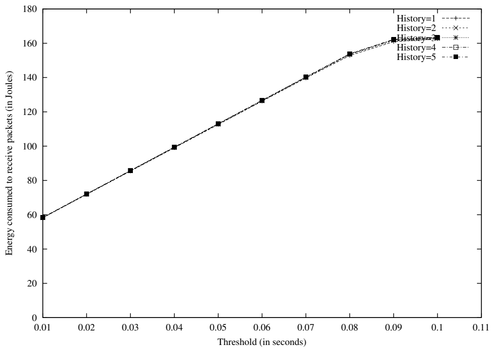
Figure 10: Duration between network packets received by the client (laptop)



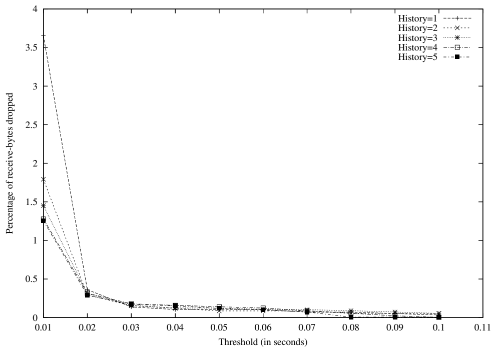
(a) Energy consumed (28.8 Kbps)



(b) Percentage bytes dropped (28.8 Kbps)



(c) Energy consumed (768 Kbps)



(d) Percentage bytes dropped (768 Kbps)

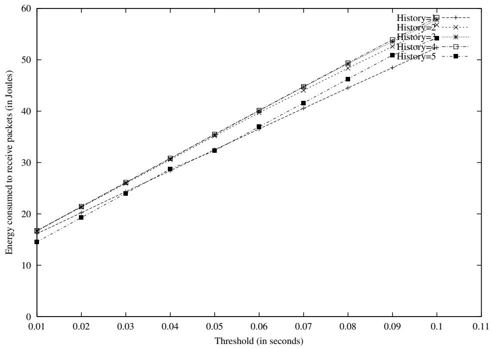
Figure 11: Microsoft media - History based scheme (0% network packet loss)

4.1.1 Network with no packet loss

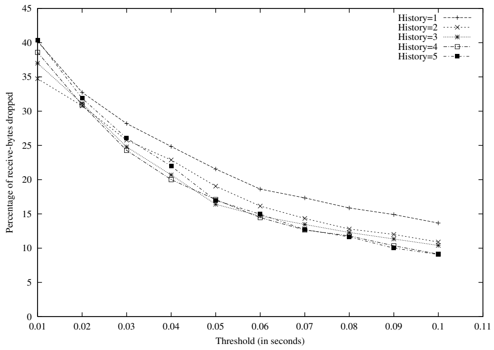
We vary the *History* and *threshold* parameters and plot the energy consumed and percentage bytes dropped (for select network bandwidths) for Microsoft media, Real and Quicktime in Figures 11, 12 and 13, respectively. The goal is to choose policies that reduce the energy consumed while losing the least amounts of data.

First we analyze the performance of history-based client-side approaches for Microsoft media streams optimized for 28.8 Kbps. We plot the total energy consumed as well the percentage of data bytes dropped in Figures 11(a) and 11(b), respectively. We note that energy consumed continues to decline as the *threshold* is reduced. Since Microsoft media tends to transmit data packets at fairly regular intervals (Section 3.1.1), the effect of *threshold* parameter is minimal; the simple history based prediction mechanisms perform acceptably. However, when the *threshold* is reduced below 0.02 seconds, there is a dramatic increase in the amount of data lost (to account for the slight variations in the times at which the packets are received). At a threshold of 0.02 second, the stream consumes 30 Joules (against 160 Joules for the traditional case) while only losing 2% of the data. Similarly, for a stream optimized for 768 Kbps (Figures 11(c) and 11(d), there is a sharp increase in loss rate as we reduce the *threshold* below 0.02 seconds. For a policy that looks at a *History* of 5 levels, a *threshold* of 0.02 seconds only requires 70 Joules of energy (as opposed to 160 Joules) while losing 0.3% of the data packets.

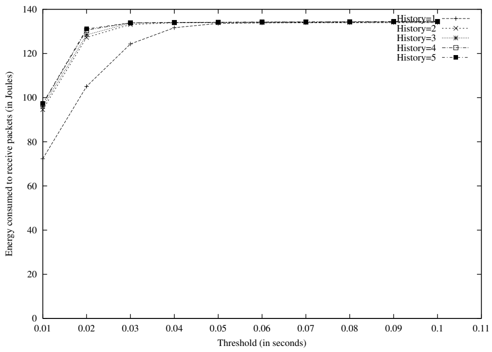
We perform similar analysis for Real streams and plot the results in Figure 12. Figure 12 shows the



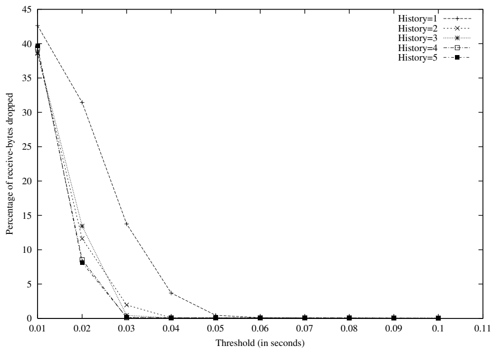
(a) Energy consumed (28.8 Kbps)



(b) Percentage bytes dropped (28.8 Kbps)

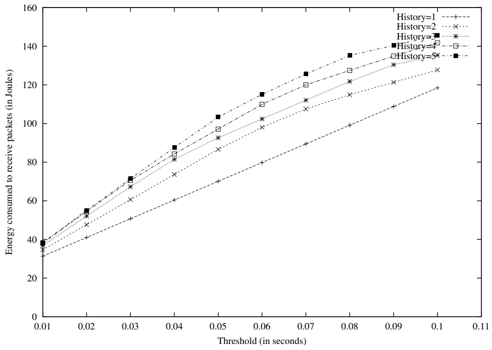


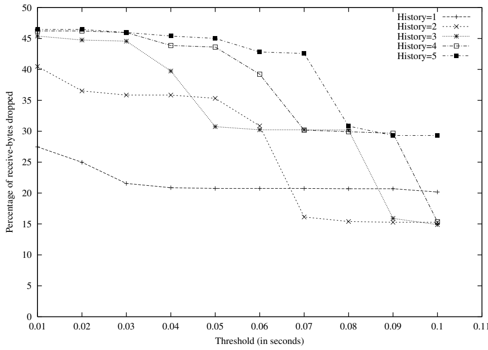
(c) Energy consumed (512 Kbps)



(d) Percentage bytes dropped (512 Kbps)

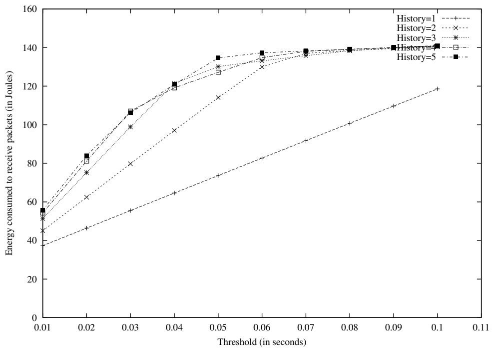
Figure 12: Real - History based scheme (0% network packet loss)

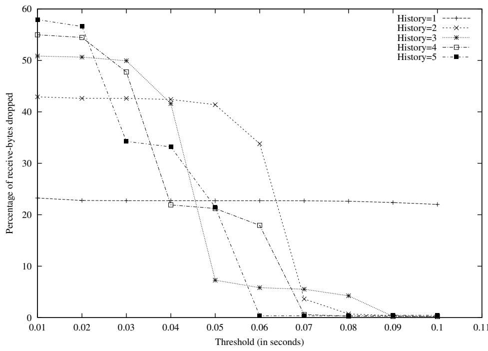




(a) Energy consumed (28.8 Kbps)

(b) Percentage bytes dropped (28.8 Kbps)





(c) Energy consumed (256 Kbps)

(d) Percentage bytes dropped (256 Kbps)

Figure 13: Quicktime - History based scheme (0% network packet loss)

tradeoff between *threshold* and cumulative energy consumption and percentage data loss. We notice that the *History* parameters has minimal influence, while the proper choice of *threshold* has a significant influence on the energy consumed and the amount of data bytes lost. For a stream optimized for 28.8 Kbps (Figures 12(a) and 12(b)), a stream that consumes 30 Joules drops over 20% of the data bytes (as opposed to 2% for Microsoft media). From Figures 12(c) and 12(d), we notice that high bandwidth Real streams offer little opportunity for energy conservation by switching to low power states. Reducing the cumulative energy consumption to 100 Joules results in a loss of over 40% of the data bytes.

Similarly, utilizing a history-based client-side approach, the total energy consumed to receive network streams as well as the amount of bytes lost for transmitting in Quicktime format at network bandwidths of 28.8 Kbps and 512 Kbps are illustrated in Figure 13. We note that a *History* parameter of 1 offers some benefit. For a stream optimized for 28.8 Kbps, a stream that consumes 40 Joules (close to the Microsoft media's 30 Joules) drops 25% of the data bytes (as opposed to 2% for Microsoft media). As noted in Section 3.3.1, Quicktime seems to perform application level fragmentation, where packets are sent in quick succession followed by long idle intervals. Such transmission patterns are harder to predict with a history based approach. Hence, Quicktime stream format is less amenable to schemes that predict the future packet arrival times based on the past arrival intervals.

We performed similar experiments for a lossy network (5% loss). In the interest of space, we do not plot the results; but note that we observed similar results to a lossless network. Microsoft media under lossy conditions exhibits less energy savings from history based approaches. Real and Quicktime performed

Stream Format	Stream b/w	Network 0% loss		Network 5% loss		*Threshold* (in sec)
		Energy (in J)	Bytes dropped (%)	Energy (in J)	Bytes dropped (%)	
Microsoft Media (*History*=1)	28.8 Kbps	30	2	30	8	0.02
	56 Kbps	35	1	36	8	0.02
	128 Kbps	52	2	52	8	0.02
	256 Kbps	60	0.5	63	6	0.02
	768 Kbps	70	0.25	81	3.5	0.02
	2000 Kbps	135	0.15	75	4.5	0.022
Real (*History*=5)	28 Kbps	55	10	100	12	0.1
	56 Kbps	116	4	130	6	0.1
	128 Kbps	82	11	60	25	0.03
	256 Kbps	120	5	150	8	0.03
	512 Kbps	132	8	43	24	0.02
Quicktime (*History*=1)	28.8 Kbps	40	25	50	28	0.02
	56 Kbps	41	30	57	30	0.02
	128 Kbps	38	38	37	26	0.02
	256 Kbps	47	23	61	20	0.02

Table 6: energy consumed and % packets dropped by the history based approach

little worse under lossy network conditions. The results for the different formats under varying network bandwidth and loss parameters are tabulated in Table 6. We note that significant energy consumption gains with minimal data loss can be expected for formats which tend to packets at a constant rate. Microsoft media tends to send packets at a constant pace and so can benefit from simple history based approaches. Quicktime tends to send packets in quick succession followed by prolonged idle intervals. Such packets are harder to predict at the network level without understanding the packet semantics. Saving energy under such network packet conditions comes at high data loss rates. On the other hand, Real packets are highly unpredictable and hence offer little possibility for history based prediction mechanisms (attenuated by high *threshold*).

5 Related Work

There has been considerable work on power management for components of a mobile device. This work includes spindown policies for disks and alternatives [4, 12, 11, 10, 16, 19, 21, 35], and managing wireless communication [17, 18, 33, 30, 9]. Agrawal et al. [2] describe techniques for processing video data for transmission under low battery power conditions. Corner et al. [8] describe the time scales of adaptation for mobile wireless video-conferencing systems. In this work, we focus our attention on receiving multimedia data.

Lorch et al. [22] present a survey of the various software techniques for energy management. Havinga et al. [14, 15] present an overview of techniques for energy management of multimedia streams.

Ellis [13] advocates high level mechanisms for power managements. Vahdat et al. [34] propose that energy as a resource should be managed by the operating system. Kravets et al. [28] advocate an end-to-end model for conserving energy for wireless communications. In our earlier work [5], we utilized transcoding as an application level technique to reduce the image data; trading off image size for network transmission and storage costs. In this work, we explore transcoding at the application level to reduce the amount of data and transitioning to lower power states at the lower levels to conserve overall energy requirements.

A number of previous works have performed detailed analysis of the behavior and access dynamics of various systems. For example, Mena et al. [23] analyzed Real audio traffic, Mogul et al. [25] analyzed web access patterns from proxy logs, Chandra et al. [6] analyzed the transcoding characteristics of web images.

Such analysis were later exploited in improving the overall system performance.

6 Conclusions

In this paper, we analyzed the network behavior and energy consumption characteristics of popular multimedia streaming formats; serving streams optimized for various network bandwidths and loss conditions. We explored Microsoft media, Real and Quicktime as the popular formats. We showed that:

- Microsoft media tends to transmit packets at fairly regular intervals. This facilitates history-based approaches to predict the arrival time of streaming packets. For high bandwidth streams, Microsoft media uses network level fragmentation. On a lossy network, such fragmentation leads to excessive packet loss (and wasted energy) as a loss of single fragment leads to a loss of the entire streaming packet. Microsoft media consumes about 160 Joules of energy at the WNIC to receive the stream.

- Real media transmits packets that are well below the MTU of the Ethernet links and so the packets are not fragmented in the network. Real stream packets tend to be sent closer to each other, especially at higher bandwidths. Real streams for a 120 second segment was transmitted in 100 seconds. This allowed the Real stream to consume less energy; Real streams consume about 120 Joules of energy at the WNIC to receive the video stream.

- Quicktime transmits packets at intervals that are less predictable than Microsoft media. Quicktime packets sometimes arrive in quick succession; most likely an application level fragmentation mechanism. Such packets are harder to predict at the network level without understanding the semantics of the packets themselves. In general, Quicktime consumes about 160 Joules of energy at the WNIC to receive the video stream.

We used these findings to develop a history-based client-side approach that utilizes the past history to predict the amount of time the WNIC spends in *sleep* state in order to conserve energy. We show that

- Microsoft media benefits immensely from a history-based client-side mechanism. For example, switching to a stream optimized for 28.8 Kbps network only consumed 30 Joules (as opposed to 160 Joules) while losing 2% of the data bytes. A higher bandwidth stream at 768 Kbps still consumes 70 Joules while losing 0.25% of the data bytes. A lossy network increases the data loss rate with similar energy savings.

- We showed that formats that send packets in less predictable fashion (Quicktime and Real) do not benefit much from a history-based client-side approach. Quicktime consumes less energy (about 40 Joules) but loses more data bytes (about 30%). Real on the other hand loses less data (about 10%) but offers little energy saving (about 100 Joules).

We believe that modifying Real and Quicktime services to transmit larger data packets at regular intervals can offer better energy consumption characteristics with minimal latency and jitter.

6.1 Future directions

We are pursuing further research in a number of directions. First, 802.11b standard allows for idle stations to go to the power saving *sleep* state. The access points buffer any packets for these sleeping stations and announce pending buffered frames as part of the periodic Beacon frame. Power saving stations can wake up prior to an expected Beacon to listen for waiting frames. In general, sleep state introduces undue jitter

Stream b/w	Microsoft Media	Real	Quicktime
28.8 Kbps	4.6315	26.7181	13.9298
56 Kbps	7.7499	32.0752	13.4824
128 Kbps	14.3066	38.5564	15.9537
256 Kbps	25.0872	50.5745	23.5908
512 Kbps	-	53.9412	-
768 Kbps	31.8450	-	-
2000 Kbps	46.9756	-	-

Table 7: Average CPU utilization (for laptop player - in %)

and delay to live streaming media transfers. However, we are exploring hybrid client-side and access point solutions to minimize the byte loss in approaches discussed in Section 4.

The overall energy consumed in watching a stream not only depends on the energy consumed by the WNIC card, but also on other system components such as the display and the CPU required to render the streams. Since the segments are of constant durations, we can expect the energy consumption of displays to be constant for the various formats and transcoding levels. We tabulate the average CPU utilization for the various formats for watching streams in various transcoding levels in Table 7. Real displays the streams in a larger window and hence tends to consume more CPU cycles for rendering. Microsoft media streams consume less CPU cycles than Quicktime. Hence, transcoding to lower quality streams not only saves WNIC energy, but also in the CPU energy consumed. We are further exploring the overall energy costs for receiving and rendering multimedia streams for various transcoding levels.

Also, battery life for typical batteries can be extended by allowing the batteries to “recover”, exploiting the battery recovery effect [7]. Our scheme of returning the network interface to the lower power *sleep* state can be expected to prolong the battery life. We are exploring the implications of recovery effect on our proposed policies.

Acknowledgments

We thank Carla Ellis for her invaluable comments and suggestions. Amin Vahdat gave important feedback on an earlier draft. Vivek Kaluskar helped us in setting up the infra-structure during the early phases. We thank David Lowenthal for his valuable comments. This work was supported in part by a research grant from the Yamacraw initiative.

References

[1] Compaq iPAQ pocket pc. http://www.compaq.com/products/handhelds/pocketpc/.

[2] P. Agrawal, S. Chen, P. Ramanathan, and K. Sivalingam. Battery power sensitive video processing in wireless networks. In *Proceedings IEEE PIMRC98*, Boston, September 1998.

[3] Apple Quicktime. http://www.apple.com/quicktime/.

[4] Mary Baker, Satoshi Asami, Etienne Deprit, John Ousterhout, and Margo Seltzer. Non-volatile Memory for Fast, Reliable File Systems. In *Proceedings of the 5th International Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 10–22, October 1992.

[5] Surendar Chandra, Carla Schlatter Ellis, and Amin Vahdat. Managing the storage and battery resources in an image capture device (digital camera) using dynamic transcoding. In *Proceedings of the*

Third ACM International Workshop on Wireless and Mobile Multimedia (WoWMoM'00), pages 73–82, Boston, August 2000. ACM SIGMOBILE.

[6] Surendar Chandra, Ashish Gehani, Carla Schlatter Ellis, and Amin Vahdat. Transcoding characteristics of web images. In Martin Kienzle and Wu chi Feng, editors, *Multimedia Computing and Networking (MMCN'01)*, volume 4312, pages 135–149, San Jose, CA, January 2001. SPIE - The International Society of Optical Engineering.

[7] Carla Fabiana Chiasserini and Ramesh R. Rao. Pulsed battery discharge in communication devices. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking (MOBICOM '99)*, pages 88–95, Seattle, WA, August 1999.

[8] Mark D. Corner, Brian D. Noble, and Kimberly M. Wasserman. Fugue: time scales of adaptation in mobile video. In *Proceedings of the SPIE Multimedia Computing and Networking Conf.*, San Jose, CA, January 2001.

[9] Anindya Datta, Aslihan Celik, Jeong Kim, Debra E. VanderMeer, and Vijay Kumar. Adaptive broadcast protocols to support power conservant retrieval by mobile users. In *Proceedings of Data Engineering Conf. (ICDE)*, pages 124–133, September 1997.

[10] Fred Douglis, Ramon Caceres, Brian Marsh, Frans Kaashoek, Kai Li, and Joshua Tauber. Storage Alternatives for Mobile Computers. In *Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI)*, pages 25–37, November 1994. Monterey, CA.

[11] Fred Douglis, P. Krishnan, and Brian Bershad. Adaptive Disk Spin-down Policies for Mobile Computers. In *2nd USENIX Symposium on Mobile and Location Independent Computing*, April 1995. Monterey CA.

[12] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the Power Hungry Disk. In *Proceedings of the 1994 Winter USENIX Conf.*, pages 293–306, January 1994.

[13] Carla S. Ellis. The case for higher-level power management. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*, Rio Rico, AZ, March 1999.

[14] Paul J. M. Havinga. *Mobile Multimedia Systems*. PhD thesis, University of Twente, February 2000.

[15] Paul J. M. Havinga and Gerard J. M. Smit. *Wireless Communications and Mobile Computing*, volume 1:2, chapter Energy-efficient wireless networking for multimedia applications, pages 165–184. Wiley, 2001.

[16] D. Helmbold, D. Long, and B. Sherrod. A Dynamic Disk Spin-Down Technique for Mobile Computing. In *Proceedings of the 2nd ACM International Conf. on Mobile Computing (MOBICOM96)*, pages 130–142, November 1996.

[17] Tomasz Imielinski, Monish Gupta, and Sarma Peyyeti. Energy Efficient Data Filtering and Communications in Mobile Wireless Computing. In *Proceedings of the Usenix Symposium on Location Dependent Computing*, April 1995.

[18] R. Kravets and P. Krishnan. Power Management Techniques for Mobile Communication. In *Proceedings of the 4th International Conf. on Mobile Computing and Networking (MOBICOM98)*, pages 157–168, October 1998.

[19] P. Krishnan, P. Long, and J. Vitter. Adaptive Disk Spin-Down via Optimal Rent-to-Buy in Probabilistic Environments. In *Proceedings of the 12th International Conf. on Machine Learning*, pages 322–330, July 1995.

[20] LAN/MAN Standards Committee of the IEEE Computer Society. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed Physical Layer Extension in the 2.4 GHz Band*. IEEE, 3 Park Avenue, New York, NY 10016, 1999.

[21] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. In *USENIX Association Winter Technical Conf. Proceedings*, pages 279–291, 1994.

[22] Jacob Lorch and Alan J. Smith. Software Strategies for Portable Computer Energy Management. *IEEE Personal Communications Magazine*, 5(3):60–73, June 1998.

[23] Art Mena and John Heidemann. An empirical study of real audio traffic. In *Proceedings of the IEEE Infocom*, pages 101–110, Tel-Aviv, Israel, March 2000. IEEE.

[24] Microsoft Windows Media Technologies. http://www.microsoft.com/windowsmedia/.

[25] Jeff Mogul and Tom M. Kroeger. Digital's web proxy traces. ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html, 1996.

[26] Packet Video. http://www.packetvideo.com/.

[27] Real Player. http://www.real.com/.

[28] Karsten Schwan Robin Kravets and Ken Calvert. Power-aware communication for mobile computers. In *International Workshop on Mobile Multimedia Communications(MoMuc'99)*, November 1999.

[29] Stephanie Sanborn. Armchair quarterbacks go wireless at 3com park. http://www.idg.net/english/crd_3com_259625.html, September 2000.

[30] S. Singh, M. Woo, and C. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of MOBICOM*, pages 181–190, October 1998.

[31] Cliff Skolnick. 802.11b community network list. http://www.toaster.net/wireless/community.html, http://www.toaster.net/wireless/aplist.php, 2001.

[32] Mark Stemm, Paul Gauthier, Daishi Harada, and Randy H. Katz. Reducing power consumption of network interfaces in hand-held devices. In *Proceedings of the 3rd International Workshop on Mobile Multimedia Communications (MoMuc-3)*, Princeton, NJ, September 1996.

[33] Mark Stemm and Randy Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices. In *Proceedings of the 3rd International Workshop on Mobile Multimedia Communications (MoMuC-3)*, September 1996.

[34] Amin Vahdat, Alvy Lebeck, and Carla Schlatter Ellis. Every joule is precious: The case for revisiting operating system design for energy efficiency. In *In Proceedings of the 9th ACM SIGOPS European Workshop*, September 2000.

[35] John Wilkes. Predictive Power Conservation. Technical Report HPL-CSP-92-5, Hewlett-Packard Labs, February 1992.