Multimedia Systems Journal manuscript No.
(will be inserted by the editor)

Wireless Network Interface Energy Consumption

Implications for Popular Streaming Formats

Surendar Chandra

Department of Computer Science, University of Georgia

Received: date / Revised version: date

Abstract With the proliferation of mobile streaming multimedia, available battery capacity constrains the end-user experience. Since streaming applications tend to be long running, wireless network interface card's (WNIC) energy consumption is particularly an acute problem. In this work, we explore various mechanisms to conserve client WNIC energy consumption for popular streaming formats such as Microsoft Windows media, Real and Apple Quicktime. First, we investigate the WNIC energy consumption characteristics for these popular multimedia streaming formats under varying stream bandwidth and network loss rates. We show that even for a high bandwidth 2000 kbps stream, the WNIC *unnecessarily* spent over 56% of the time in *idle* state; illustrating the potential for significant energy savings.

Based on these observations, we explore two mechanisms to conserve the client WNIC energy consumption. First we show the limitations of IEEE 802.11 power saving mode for such isochronous multimedia streams. Without an understanding of the stream requirements, these scheduled rendezvous mechanisms do not offer any energy savings for multimedia streams over 56 kbps. We also develop history-based client-side strategies to reduce the energy consumed by transitioning the WNICs to a lower power consuming *sleep* state. We show that streams 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. We also show that Real and Quicktime 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. Multimedia players for popular streaming formats such as Microsoft media [23], Real [24] and Quicktime [2] are freely available for commodity mobile devices such as laptops and PDAs (e.g. Compaq iPAQ). Public venues [27] are also rapidly deploying high speed IEEE 802.11b [19] based public wireless LAN networks.

As the technologies to create and deliver mobile multimedia streams to mobile devices mature, battery capacity plays an important role in defining the end-user experience. For example, a typical PDA such as the iPAQ (equipped with two 2850 mWh batteries, one each in the unit and the PCMCIA sleeve) consumes 929 mW while fully operational [9, 4]. System components such as wireless network interface cards (WNIC), displays and CPUs consume significant amounts of power in mobile devices. For example, a 2.4 GHz IEEE 802.11b Wavelan card [29, 13] alone consumes 1425 mW for receiving data and 1675 mW for transmitting data. A necessary feature for mass acceptance of a streaming multimedia device is acceptable battery life. Future trends in battery technologies alone (along with the continual pressure for further device miniaturization) do not promise dramatic improvements that will make this issue disappear.

Since streaming applications tend to be long running, WNIC energy consumption is particularly an acute problem. Hence, it is important to look at techniques to reduce the energy consumed by the network interface to download the stream. Typically, system components can operate in several hardware states, each with their own power consumption characteristics. For example, the 2.4 GHz IEEE 802.11b Wavelan card consumes 177 mW while in *sleep* state, but consumes 1319 mW while *idle*. If the WNIC stays in *idle* state (instead of *sleep* state) while waiting for data packets, the overall energy consumed can be expected to be close to the energy consumed while actually receiving data [29]. Tremendous advances in hardware technologies for lower power devices must be matched with software schemes that reduce the amount of time spent in high-power consuming states. Frequent transitions to lower power consuming states can also be expected to allow the batteries to exploit the battery "recovery" effect [3] to prolong battery life.

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 the amount of data can also be expected to decrease 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 client WNIC energy consumption.

Our primary goal in this work was to reduce the energy required to consume a certain multimedia stream (of a given quality); not to reduce the overall energy consumption by always watching a lower quality stream. We designed our system to consume energy proportional to the stream quality. 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.

First we investigate the energy consumption implications of changing the fidelity of popular multimedia streaming mechanisms under varying network conditions. For our experiments, we explored Microsoft media [23], Real [24] and

Quicktime [2] as popular streaming formats. 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 while 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. 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. Overall, the WNIC spent most of the time waiting for stream packets in higher power consuming *idle* state. Even for a high bandwidth 2000 kbps stream, the WNIC *unnecessarily* spent over 56% of the time in *idle* state; illustrating the potential for significant energy savings.

Based on these observations, we explore two mechanisms to conserve the client WNIC energy consumption. First, we explored the effectiveness of IEEE 802.11 power saving modes of operation. We show the limitations of MAC level IEEE 802.11 power saving mode for isochronous multimedia streams. The access points have to balance potential energy savings for a single mobile client with a need for fair allocation of network resources. Without an understanding of the stream requirements, these MAC level mechanisms do not offer any energy savings for multimedia streams over 56 kbps.

Next, based on our observations about the stream behavior, we developed history based client-side techniques to exploit the stream behavior and lower the energy required to receive these streams. 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. For high bandwidth streams, Microsoft media exploits network level fragmentation, which can lead to excessive packet loss (and wasted energy) in a lossy network. 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 analyzes the effectiveness of IEEE 802.11 power management scheme to conserve energy for our streams. Section 5 explores client side policies to reduce the energy required to receive these multimedia streams. Section 6 describes related work with conclusions in Section 7.

2 Experiment Objectives and Design

2.1 Objectives

Our primary goal was to reduce the energy required by the wireless client to consume a certain multimedia stream (of a given quality). 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 realize energy savings at the network MAC level without an understanding of the application level stream dynamics? and,
- Can we develop client-side strategies to reduce energy consumption?

We analyze the energy savings for the multimedia player alone. In general, client-side policies require minimal modifications to the streaming server.

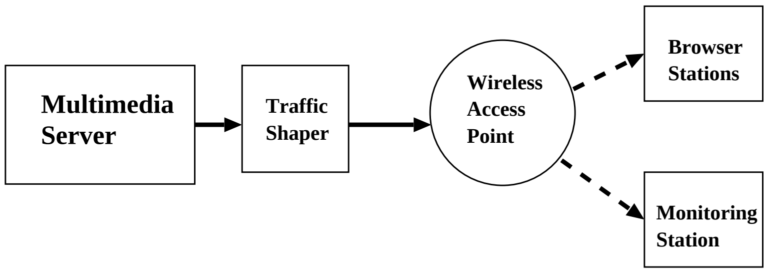


Fig. 1 Experiment Setup

2.2 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 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).
- **Traffic Shaper**: The various network conditions were simulated using a dummynet [25] based FreeBSD (http://www.freebsd.org/) 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 used a dedicated D-Link DWL 1000, Orinoco RG 1000 and Orinoco AP 500 access points. The AP 500 was connected to an external range extender antenna. Throughout our experiments, we had turned off the security encryption feature of the access points.
- **Browser Stations**: Since the fraction of WNIC energy consumption in a PDA device is significantly more than a laptop device, we expect our results to have a significant impact on an iPAQ device. We experimented with a Compaq iPAQ 3650 Pocket PC with 32 MB RAM running Windows CE 3.0 SP1 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. Since the packet characteristics for the iPAQ were similar

to the packets to the laptop player and since he Quicktime and Real players were not available for the iPAQ, we only discuss results from the laptop player. 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.

– **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).

2.3 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 was 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 publicly available streams (of lower quality settings). In the interest of space, we only discuss this video stream for the rest of this paper.

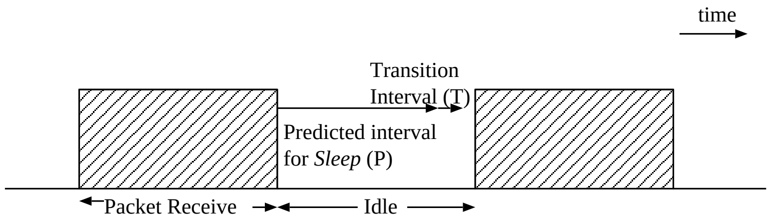


Fig. 2 Stream network packet states (simplified)

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 ([13,29]) for a Wavelan 2.4 GHz (11 Mbps) WNIC card: 177 mW - *sleep* state, 1319 mW - *idle* state, 1425 mW - receiving state and 1675 mW - transmit state. We assumed that the wireless network provides an 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 *PredictedSleepInterval* = $\frac{\sum_{History} PastIdleTimes}{History}$ - *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.

Table 1 Amount of data received

Stream Format	**Stream bandwidth**	**Transmit** (KB)	**Receive** (MB)
Microsoft Media	28.8 kbps	4.1	0.27
	56 kbps	7.1	0.40
	128 kbps	3.9	1.49
	256 kbps	4.1	3.29
	768 kbps	4.1	8.68
	2 Mbps	4.2	26.30
Real	28 kbps	10.8	0.34
	56 kbps	10.2	0.57
	128 kbps	9.9	1.29
	256 kbps	10.0	3.50
	512 kbps	10.9	6.83
Quicktime	28.8 kbps	64.99	0.47
	56 kbps	60.4	0.62
	128 kbps	63.0	1.24
	256 kbps	75.9	3.65

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 mispredictions. 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 Table 1. Overall, we note that the three formats send comparable amounts of multimedia data

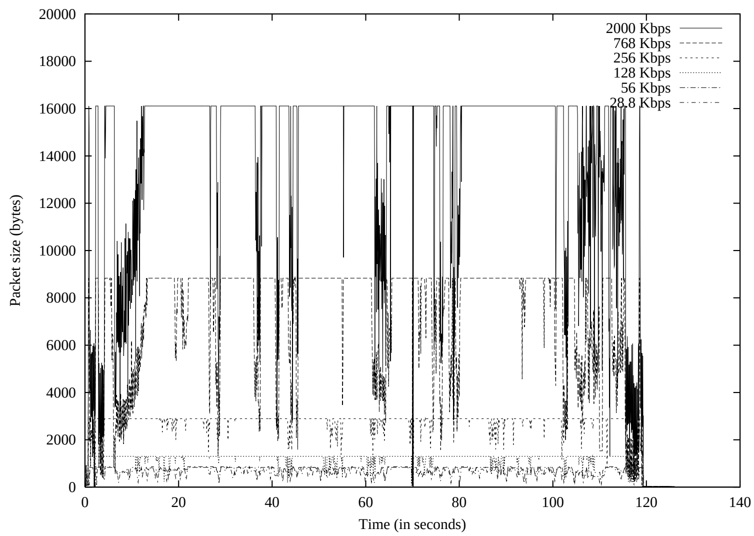


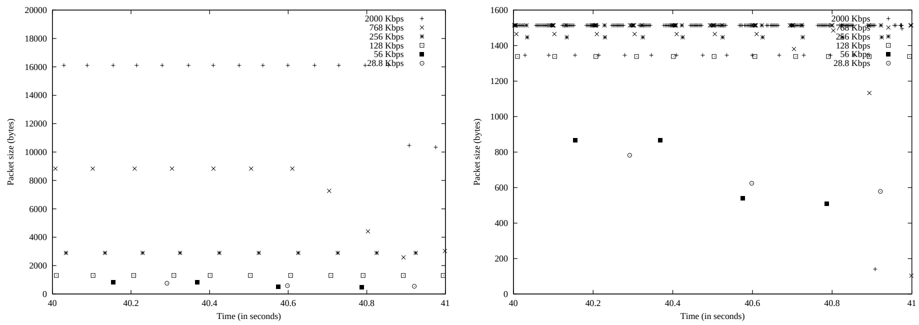
Fig. 3 Microsoft Media - Packet lengths to client on a network that does not drop any packets

to the client. However, Quicktime utilizes more feedback bandwidth compared to Real or Microsoft media. Quicktime also uses multiple UDP ports to transfer audio and video information independently. The Microsoft media player on the iPAQ utilized more feedback than the player on the laptop; presumably because the iPAQ had a harder time keeping up with the stream.

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



(a) Packet lengths (closeup) (b) Network packet lengths (closeup)

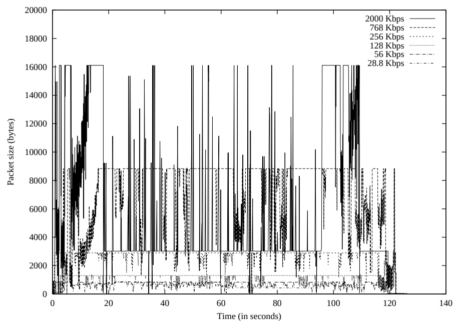
Fig. 4 Microsoft Media - Closeup view of packets to client from 40:00 through 41:00 seconds since start of stream

so, the streams progressively adapted to consume the slated overall network bandwidth.

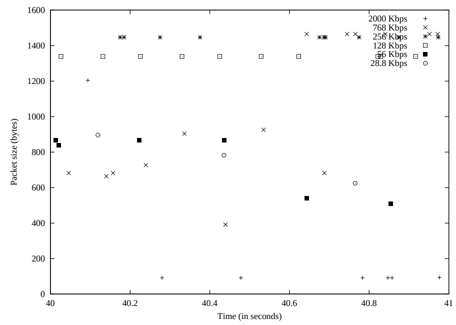
We note that once the streams stabilize, they request packets of similar lengths; higher quality streams result in larger packet sizes. Streams optimized for 28.8 kbps, 56 kbps, 128 kbps, 256 kbps, 768 kbps and 2 Mbps request packets of sizes 862 bytes, 834 bytes, 1305 bytes, 2893 bytes, 8831 bytes and 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 packets tends to be received at seemingly constant intervals. Such constant packet intervals can assist us in developing client-side policies to transition the WNIC to *sleep* 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 received 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



(a) Packet lengths



(b) Network packet lengths (closeup)

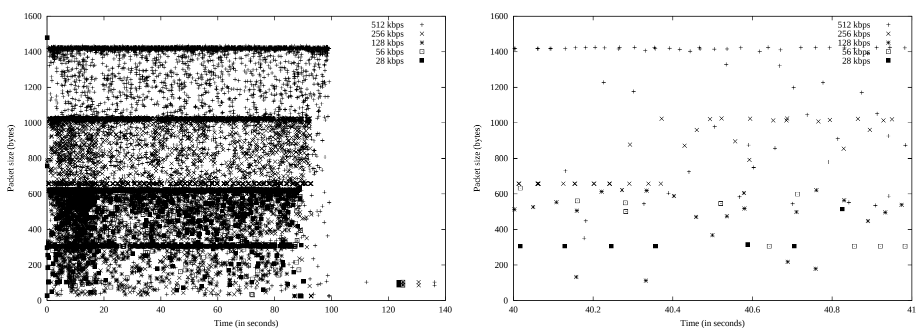
Fig. 5 Microsoft Media - Client on network with 5% random packet loss

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 sender initiated changes can adversely affect any adaptation policies explored in Section 5. 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 received (from top to the bottom). This effect was also noted by Stemm et al. [29]

Table 2 Energy consumed by the WNIC for receiving multimedia streams (in Joules)

Stream Format	Stream bandwidth	Energy consumed (in Joules)	
		Network - 0% loss	Network - 5% loss
Microsoft Media	28.8 kbps	157.50	157.69
	56 kbps	157.56	157.63
	128 kbps	157.71	159.11
	256 kbps	160.24	159.23
	768 kbps	163.23	164.19
	2 Mbps	174.79	163.92
Real	28.8 kbps	119.24	136.07
	56 kbps	119.15	136.23
	128 kbps	119.20	174.65
	256 kbps	124.31	156.29
	512 kbps	133.23	160.20
Quicktime	28.8 kbps	149.66	149.97
	56 kbps	149.74	150.01
	128 kbps	150.08	149.86
	256 kbps	149.24	152.88



(a) Packet lengths (plotted using a scatter plot for clarity)

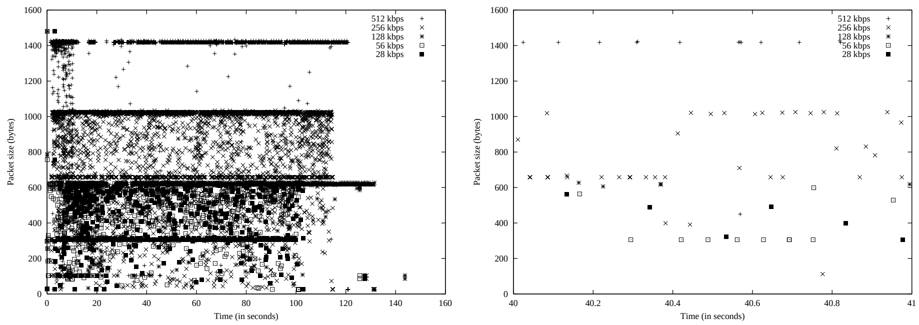
(b) Packet lengths (closeup)

Fig. 6 Real - Packet lengths to client on a network with 0% packet loss

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 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



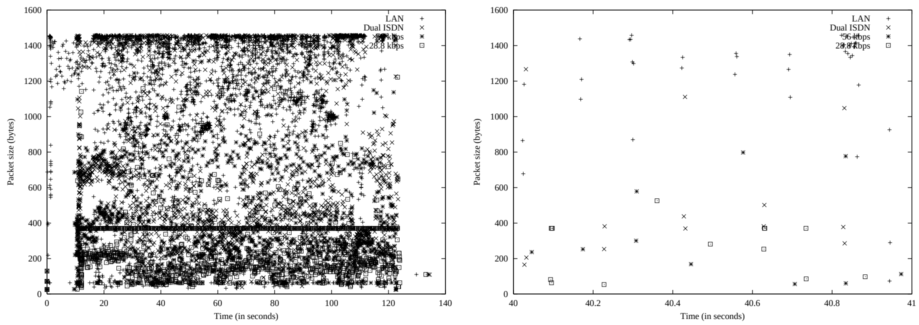
(a) Packet lengths (plotted using a scatter plot for clarity)

(b) Packet lengths (closeup)

Fig. 7 Real - Packet lengths to client on network with 5% packet loss

player was configured to turn off all local buffering (to prevent the players from *reading ahead* into the stream). 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 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 received 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



(a) Packet lengths (plotted using a scatter plot for clarity)

(b) Packet lengths (closeup)

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

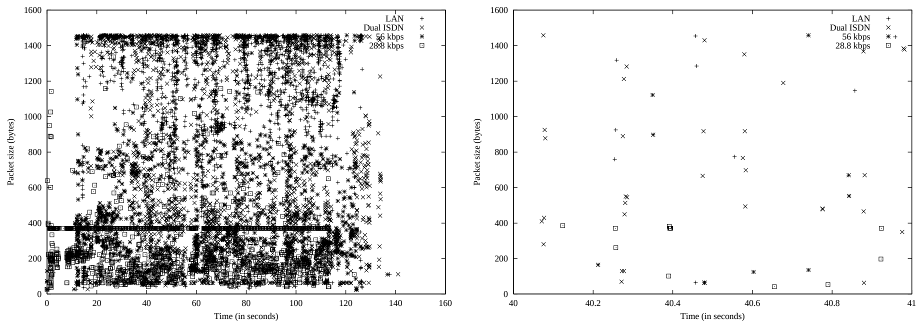
stream packets are received less often than for a loss-less network, indicating an adaptation to a lower quality stream.

3.2.3 Energy consumed by the WNIC interface The cumulative energy consumed for the various Real streams are tabulated in Table 2. As discussed earlier (Section 3.2.1), the entire Real stream was received 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 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 by the laptop with time. A closer look at the size of packets received in the



(a) Packet lengths (plotted using a scatter plot for clarity)

(b) Packet lengths (closeup)

Fig. 9 Quicktime - Packet lengths to client on network with 5% packet loss

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 2. 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).

Table 3 Idle times for the various streaming formats

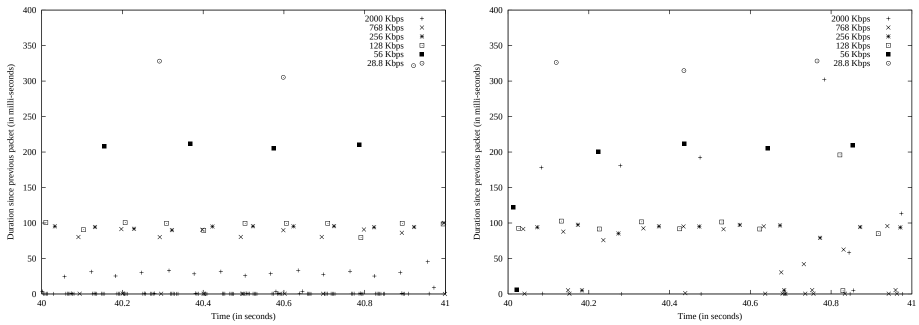
Stream Format	Stream bandwidth	Network (0% pkt loss)		Network (5% pkt loss)	
		Idle	**Receive**	**Idle**	**Receive**
Microsoft Media	28.8 kbps	122.08	0.55	122.72	0.49
	56 kbps	118.73	0.83	122.23	0.84
	128 kbps	116.38	3.05	117.49	3.06
	256 kbps	116.28	6.75	115.19	7.14
	768 kbps	104.73	17.78	97.142	21.88
	2000 kbps	69.62	53.88	102.95	19.79
Real	28 kbps	89.69	0.72	102.44	0.73
	56 kbps	89.14	1.19	102.00	1.19
	128 kbps	87.60	2.66	131.40	1.04
	256 kbps	86.65	7.20	110.89	7.20
	512 kbps	109.97	14.01	116.72	4.53
Quicktime	28.8 kbps	138.09	1.10	127.60	1.14
	56 kbps	132.30	1.40	134.80	1.44
	128 kbps	135.92	2.69	138.47	2.50
	256 kbps	125.67	6.76	148.90	5.48

3.4 Operating system influence

The server operating system, with its unique scheduling characteristics might affect the transmission characteristics for the various multimedia services. We performed experiments to identify the effects of the server operating system on the stream reception characteristics. We streamed the Apple Quicktime streams from a Dual 933 MHz Pentium Xeon machine running FreeBSD 4.5 (STABLE). We verified that the client stream reception characteristics were similar to the streams broadcast from the Windows 2000 Server. We did not have access to Real and Microsoft media servers for other alternative operating systems, but fell confident that the operating systems do not significantly affect the stream transmissions.

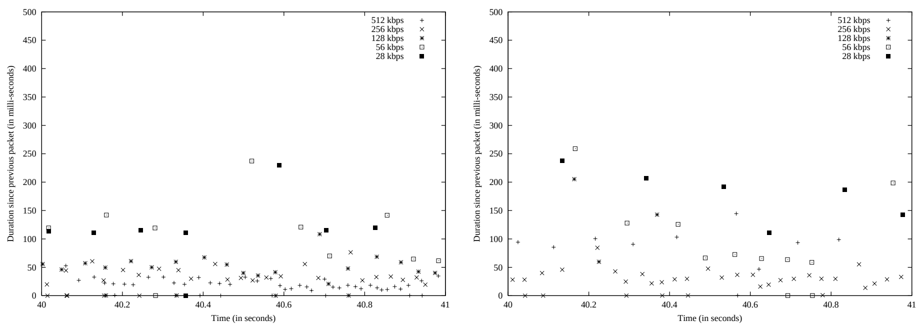
3.5 Energy saving potential

Before we investigate energy consumption reduction strategies, first we tabulate the overall idle slots in our stream traces in Table 3. We note that for low quality streams, the network interface is idle for almost all the time. Even for high quality



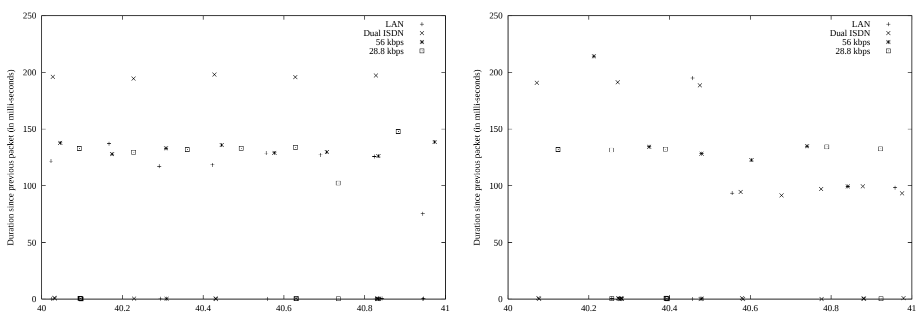
(a) 0% packet loss (b) 5% packet loss

Fig. 10 Duration between network packets received by the client (Microsoft media)



(a) 0% packet loss (b) 5% packet loss

Fig. 11 Duration between network packets received by the client (Real)



(a) 0% packet loss (b) 5% packet loss

Fig. 12 Duration between network packets received by the client (Quicktime)

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.

To summarize, in this section we analyzed the energy implications of receiving network packets for the various streaming formats. We showed the potential for significant energy savings. We noted that Microsoft media packets transmitted

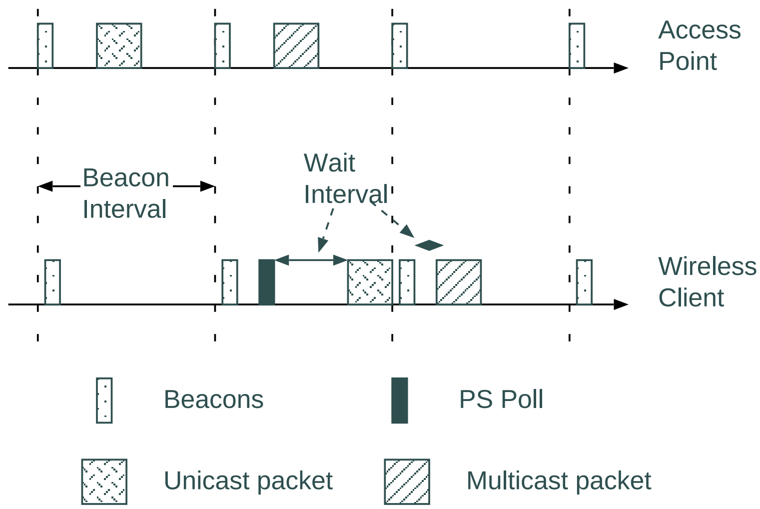


Fig. 13 IEEE 802.11 Power Saving Mode (simplified)

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 received 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 Implications of IEEE 802.11 Power Management in Wireless Access Points

In the last section, we showed the potential for significant energy savings. In this section, we explore the effectiveness of IEEE 802.11 MAC level power saving mode for conserving the client WNIC energy consumption. In the next section, we investigate a client side mechanism to predict the packet arrival times in order to effect a transition the WNIC to a lower power consuming state to conserve energy.

The IEEE 802.11 wireless LAN access standard [18] defines a power saving mode of operation wherein the wireless station and the access point cooperates

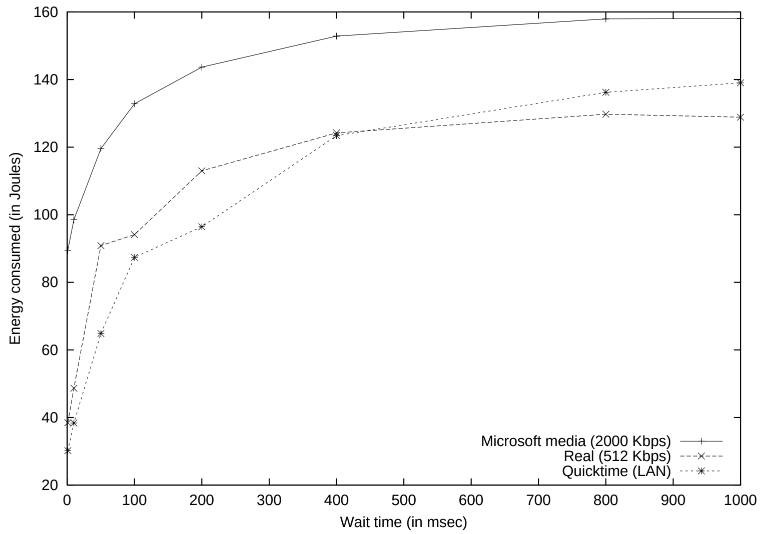


Fig. 14 Energy consumed for varying *Wait* times

to conserve energy (illustrated in Figure 13). The wireless station informs the access point of its intention to switch to power saving mode. Once in power saving mode, the wireless station switches the WNIC cards to a lower power *sleep* state; periodically waking up to receive beacons from the access point. The access point buffers any packets for this station and indicates a pending packet using a traffic indication map (TIM). These TIMs are included within beacons that are periodically transmitted from the access point. On receipt of an indication of a waiting packet at the access point, a wireless client sends a PS poll frame to the access point and waits for a response in the active (higher energy consuming) state. The access point responds to the poll by transmitting the pending packet or indication for future transmission. The access point indicates the availability of multiple buffered packets using the *More* data field. Note that the standard does not define a power saving transmit mode for the wireless station itself, it can transmit a packet whenever it wants (regardless of the beacon).

At first glance, it would appear that the 802.11 power saving mode can indeed allow the wireless clients to conserve energy by allowing them to frequently transition to lower power consuming *sleep* state. However, the potential energy saving depends on minimal *Wait* interval (time between the transmission of PS poll message and receipt of the data packet; illustrated in Figure 13). The IEEE 802.11 standard does not specify any time bound for this *Wait* interval. For a general purpose wireless access point, reducing the *Wait* interval has the inadvertent side effect of increasing the priority of data packets for wireless stations operating in the power saving mode. Access point manufacturers typically associate power saving mode as a lower throughput state.

In order to understand the implications of this wait interval, we developed a energy simulator that modeled the various power saving states of the 2.4 GHz Lucent wireless card [28,13]. We chose a beacon interval of 100 ms (used by Orinoco access points) and varied the average *Wait* times for reasonable values of 0 through 1000 msec. The access points are modeled with infinite buffer space; as the wait times increase, more packets are buffered and are sent back-to-back in succession. A realistic access point would drop these packets once the buffers fill up.

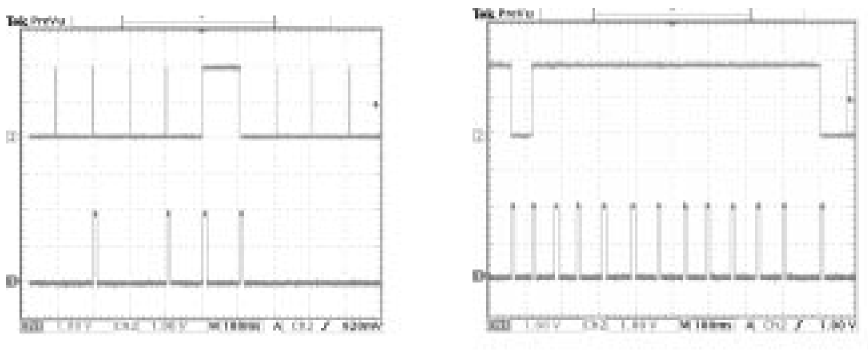
We plot the results for varying the average *Wait* interval for our reference streams. To receive these streams, the WNICs needed to be in active *read* state for 43.63%, 11.30% and 5.11% of the time, which corresponds to a necessary energy consumption of 90.24, 42.94 and 33.57 Joules, respectively. From Figure 14, we note that the potential energy saving is heavily dependent on the average *Wait* intervals. Wait times of zero illustrates the necessary energy consumption values. For average wait times less than 100 msec, even small increase in average *Waits* can drastically affect the energy saving. However, larger wait times do not offer much energy savings.

Hence, we tried to measure the actual *Wait* times for typical access points. We carefully disassembled the plastic shielding around a Orinoco Silver PC card and hooked two voltage probes around the two status LEDs. The first LED showed the card power state (transitioning to a high voltage stage while the card is active) while a second LED showed when a data packet was transmitted or received. We connected the PC card to an external Orinoco range extender antenna to reduce the effects of our instrumentation on the proper operation of the wireless card.

We used this instrumented PC WNIC card on laptops running MS Win 2000, Win 98, Redhat Linux 7.2 and Compaq iPAQ; configured the PC card to operate in power saving mode and watched media streams using MS Media, Real and Apple Quicktime players for the Windows machines, Real for Linux and PocketTV and MS Media players for the iPAQ. For our study, we used Orinoco RG 1000, Orinoco AP 500 and D-Link DWL 1000 access points. All the access points were set up on a dedicated LAN segment (with no background network traffic) and operating on the same wireless channel.

We noticed that Orinoco RG 1000 and AP 500 access points used a beacon interval of 100 msec (in spite of our attempts at changing this interval; the Linux drivers provides an API to request a different beacon interval). The D-Link DWL 1000 allowed us to choose an interval of either 160 msec or 80 msec; the Windows driver did not allow us to modify this interval and chose 80 msec for TIM. We plot several representative results for the card status during our experiment in Figure 15. Note that the different plots show different parts of the video stream. We show the results for several low bitrate streams. These lower quality streams transmit less date and can be expected to offer considerable energy savings. In each graph, the plot for Channel 2 (top) shows the duration that the card stays in active state. Channel 1 (bottom) shows the duration when an actual packet is either transmitted or received at the card. Ideally, we want the top plot to be active for the least amount of time; appearing similar to the bottom plot.

Figure 15(a) plots the results for viewing a MS stream at 56 kbps using the iPAQ device. From Figure 15(a), we note that the power save mode sometimes works optimally. The first two data packets were received with the card in higher power state for the least amount of time. The third packet however triggers the card to stay in higher energy consuming state for a long time, the access point does not transmit the next packet until the next beacon interval (a *Wait* interval of 100 msec). Also in Figure 15(b), we notice longer wait intervals for watching a Real stream at 56 kbps. As noted in Section 3, Real tends to spend many smaller packets



(a) iPAQ, MS Media stream at 56 kbps, Orinoco AP500 Access point

(b) Windows 2000, Real stream at 56 kbps, Orinoco AP500 Access point

Fig. 15 Status of IEEE 802.11b wireless PC card in powersave mode (the top plot (Ch 2) shows the WNIC power state and the bottom plot (Ch 1) shows data transmission/reception intervals. The x-axis shows the time in 100 msec ticks with voltage along the y-axis)

(as compared to Microsoft media). Such packets tend to leave the WNIC at higher energy consuming active state while the access points tries to operate "fairly" for a general audience. In fact, we noticed that watching any stream over 56 kbps tends to completely leave the WNIC in higher energy consuming active state (even though it must be possible to keep the *Wait* times lower, albeit consuming most of the available bandwidth).

In general, we believe that the 802.11b power saving mode has limitations for saving client WNIC energy consumption for the following reasons:

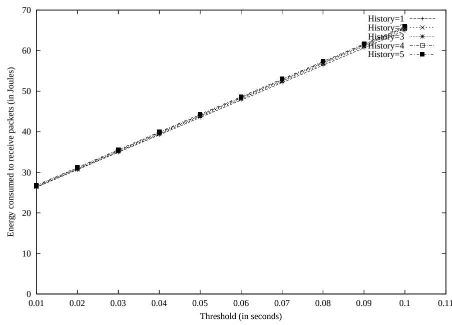
1. **Power save operation not application aware:** With the $notification(TIM) \rightarrow request(PSpoll) \rightarrow datatransmission$ model of operation, the IEEE 802.11b standard assumes that the data traffic is sporadic and well behaved (few packets spread evenly). On the other hand, multimedia streams tend to be isochronous. Formats such as Real tends to transmit smaller packets at close intervals. Formats such as MS media tends to utilize network fragmentation leading to fragmented packets sent closer to each other. Delaying parts of a fragment can delay the delivery of the entire packet to the multimedia browser.
2. **Access point behavior hardware dependent:** The potential energy saving depends on the policy choices at both the access point and the mobile client

station. A general access point needs to balance the need for power saving on a single client with fairly sharing the available bandwidth. Even with no other clients to share the bandwidth, the access points tested still tended to keep the clients waiting for extended periods of time.

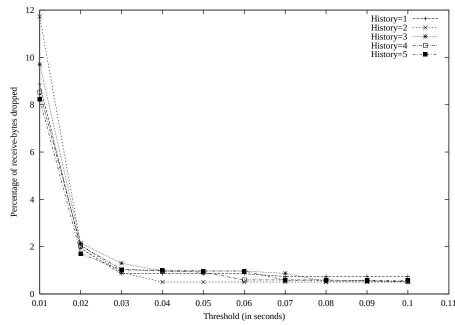
5 Strategies for adapting client network interface to lower power state

In Section 3, we analyzed the packet reception behavior for the various streaming formats under changing network conditions. In the previous section, we discussed the limitations of utilizing the 802.11b MAC power management scheme for popular streaming media formats. 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. Our goal is to develop policies that allow the network interface to enter the *sleep* state as much as possible; without losing network packets.

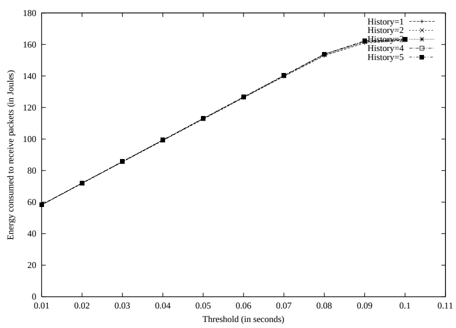
In order to develop such transition strategies, first we plot the inter-packet arrival times for the various streaming formats under changing network conditions in Figures 10,11 and 12. In Figure 10(a), we note the presence of fragmented network packets for Microsoft media streams (consecutive packets 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 12(a). High quality Real stream sends packets with small inter-packet arrival times (less than 50 msec) (Figure 11(a)). 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.



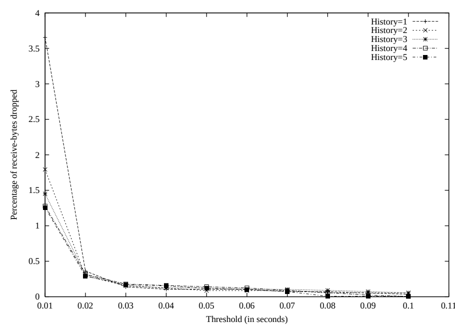
(a) Energy consumed (28.8 kbps)



(b) Percentage bytes dropped (28.8 kbps)



(c) Energy consumed (768 kbps)



(d) Percentage bytes dropped (768 kbps)

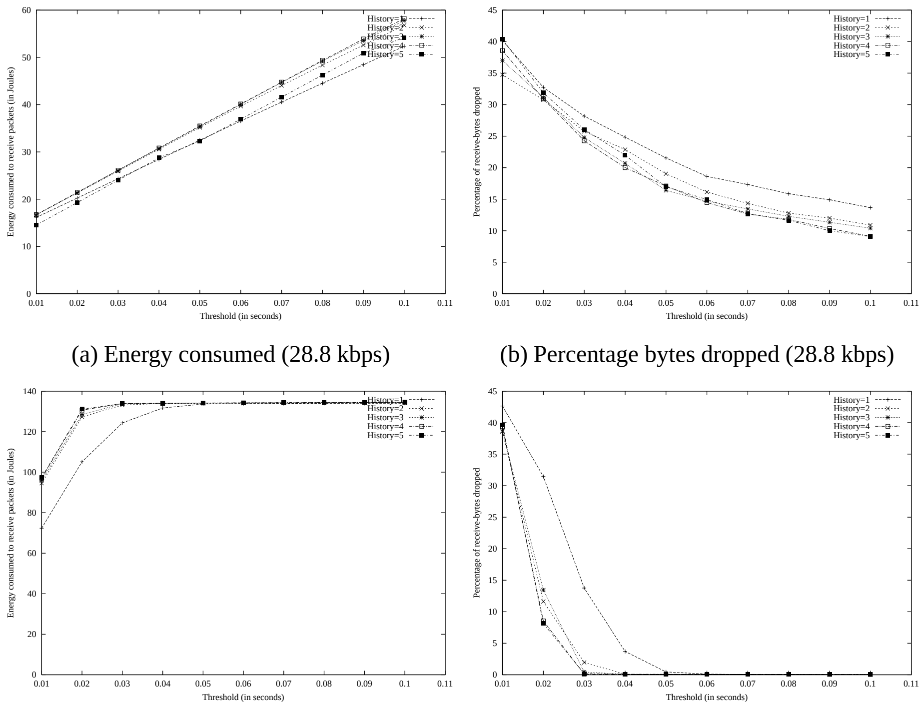
Fig. 16 Microsoft media - History based scheme (0% network packet loss)

5.1 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 $Predicted\ sleep\ interval = \frac{\sum_{History} Past\ idle\ times}{History} - threshold.$

5.1.1 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 16, 17 and 18, 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 16(a) and 16(b),



(a) Energy consumed (28.8 kbps)

(b) Percentage bytes dropped (28.8 kbps)

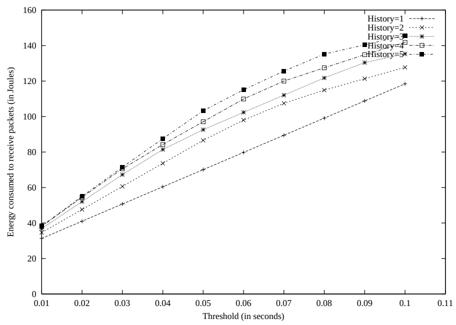
(c) Energy consumed (512 kbps)

(d) Percentage bytes dropped (512 kbps)

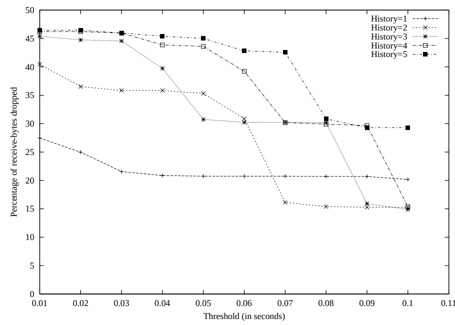
Fig. 17 Real - History based scheme (0% network packet loss)

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 16(c) and 16(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.

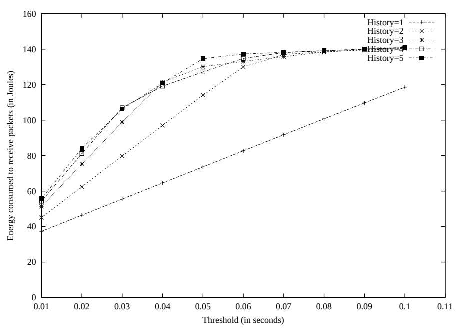
We perform similar analysis for Real streams and plot the results in Figure 17. Figure 17 shows the trade off between *threshold* and cumulative energy consump-



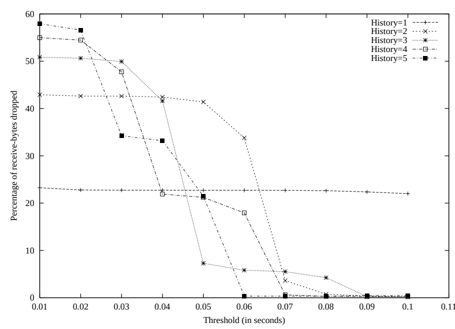
(a) Energy consumed (28.8 kbps)



(b) Percentage bytes dropped (28.8 kbps)



(c) Energy consumed (256 kbps)



(d) Percentage bytes dropped (256 kbps)

Fig. 18 Quicktime - History based scheme (0% network packet loss)

tion and percentage data loss. We notice that the *History* parameters have 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 17(a) and 17(b)), a stream that consumes 30 Joules drops over 20% of the data bytes (as opposed to 2% for Microsoft media). From Figures 17(c) and 17(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 18. 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 little worse under lossy network conditions. The results for the different formats under varying network bandwidth and loss parameters are tabulated in Table 4. 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*).

6 Related Work

There has been considerable work on power management for components of a mobile device. This work includes spin-down policies for disks and alternatives [32, 20, 7, 14], scheduling policies for reducing CPU energy consumption [31, 12] and managing wireless communications [15, 6, 16, 26]. A number of previous works have performed detailed analysis of the behavior and access dynamics of various systems. For example, Mena et al. [22] analyzed Real audio traffic. Such analysis were later exploited in improving the overall system performance.

Table 4 energy consumed and % packets dropped by the history based approach

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

Our work is similar in spirit to the work of Feeney et al. [10]. They obtain detailed measurements of the energy consumption of an IEEE 802.11 wireless network interface operating in an ad hoc networking environment. They showed that the energy consumption of an IEEE 802.11 wireless interface has a complex range of behavior and that the energy consumption was not synonymous with bandwidth utilization. Our work explores similar techniques for WNIC energy management for multimedia traffic.

Lorch et al. [21] presented a survey of the various software techniques for energy management. Havinga et al. [13] presented an overview of techniques for energy management of multimedia streams. Agrawal et al. [1] described techniques for processing video data for transmission under low battery power conditions. Corner et al. [5] described the time scales of adaptation for mobile wireless videoconferencing systems.

Ellis [8] advocates high level mechanisms for power management. Flinn et al. [11] demonstrated such a collaborative relationship between the operating system and application to meet user-specified goals for battery duration. Vahdat et al. [30]

proposed that energy as a resource should be managed by the operating system. Kravets et al. [17] advocated an end-to-end model for conserving energy for wireless communications. In this work, we explore high level mechanisms to enable the mobile client to transition to lower power states and reduce overall energy requirements.

7 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 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 received 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 show the limitations of IEEE 802.11 power saving mode for receiving popular multimedia streams (Microsoft media, Real and Quicktime). We showed that the non-deterministic TIM interval and the associated *Wait* interval can adversely affect the potential energy savings. We showed that the WNICs effectively switch out of the power saving modes for even moderately high bandwidth streams (128 kbps). Also, a single TIM can reduce the energy savings for multiple clients contending for the same TIM beacons by increasing the wait intervals for each 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.

Acknowledgements We thank Ben Bishop for his help with the oscilloscope. This work was supported in part by a research grant from the Yamacraw initiative.

References

1. P. Agrawal, S. Chen, P. Ramanathan, and K. Sivalingam. Battery power sensitive video processing in wireless networks. In *Proceedings IEEE PIMRC98*, Boston, September 1998.
2. Apple Quicktime. http://www.apple.com/quicktime/.
3. 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.
4. Sukjar Cho. Power Management of iPAQ, February 2001.
5. 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.
6. 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.
7. 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.
8. 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.
9. Keith I. Farkas, Jason Flinn, Godmar Back, Dirk Grunwald, and Jennifer M. Anderson. Quantifying the Energy Consumption of a Pocket Computer and a Java Virtual Machine. In *ACM SIGMETRICS*, pages 252–263, Santa Clara, CA, June 2000.
10. Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings IEEE INFOCOM 2001*, volume 3, pages 1548–1557, Anchorage, Alaska, April 2001.
11. Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Symposium on Operating Systems Principles (SOSP)*, pages 48–63, December 1999.

12. Kinshuk Govil, Edwin Chan, and Hal Wasserman. Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU. In *Proceedings of 1st ACM International Conference on Mobile Computing and Networking (MOBICOM95)*, pages 13–25, November 1995.

13. Paul J. M. Havinga. *Mobile Multimedia Systems*. PhD thesis, Univ. of Twente, February 2000.

14. 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.

15. 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.

16. 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.

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

18. LAN/MAN Standards Committee of the IEEE Computer Society. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE, 3 Park Avenue, New York, NY 10016, 1999.

19. 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.

20. 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.

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

22. 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.

23. Microsoft Windows Media Technologies. http://www.microsoft.com/windowsmedia/.

24. Real Player. http://www.real.com/.

25. Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, January 1997.

26. Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-Aware Routing in Mobile Ad Hoc Networks. In *The Fourth Annual ACM/IEEE International Conf. on Mobile Computing and Networking*, pages 181–190, 1998.

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

28. 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.

29. Mark Stemm and Randy H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. In *IEICE (Institute of Electronics, Information and Communication Engineers) Transactions on Communications, special Issue on Mobile Computing*, volume E80-B, pages 1125–1131, August 1997.

30. 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.

31. Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for Reduced CPU Energy. In *USENIX Association, Proceedings of First Symposium on Operating Systems Design and Implementation (OSDI)*, November 1994. Monterey CA.

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