

Content-aware Power Optimizations for Multimedia Streaming Over Wireless Networks

Radu Cornea Alex Nicolau Nikil Dutt
Donald Bren School of Information and Computer Science
University of California, Irvine, CA 92697-3425
{radu,nicolau,dutt}@ics.uci.edu

CECS Technical Report #06-13

November 2006

Abstract

The relatively high power consumption of wireless network interfaces represents an important detriment in multimedia streaming for mobile devices. The IEEE 802.11 built-in power saving mode was designed for transfers of different nature and is not able to take advantage of the short idle intervals and continuous, periodic transmissions inherent in multimedia streaming. We propose an annotation based approach to wireless network power management that analyzes the variations in data transfer bandwidth during playback and uses the results to buffer transmissions into larger burst transmissions with longer idle periods when the network card is transitioned into a lower power, sleep mode. Annotations allow for energy savings of up to 75% for the network interface, with practically no quality degradation or packet loss, only a small delay due to the buffer.

1 Introduction

Recent deployments of metropolitan scale wireless access in various cities and the perspective of future proliferation of free wireless hotspots, have created tremendous opportunities for streaming multimedia content over the wireless network. At the same time, we are witnessing an increased adoption of wireless technology in portable devices: PDAs, cell phones, but also more recently mp3 and video players. The main drawback of this technology is the limited battery life associated with wireless enabled devices. These devices are increasingly being used in content (multimedia) delivery over the network.

However, battery technology does not match the advances afore mentioned, remaining a limiting factor of operating life in portable devices. The main power consuming components of a handheld device are the CPU, display and network interface. The wireless network interface (card) accounts for a large if not the largest percentage of battery power in such devices (up to 50% for handheld devices and slightly less for laptops). Multimedia applications further aggravates the situation, being both CPU and network intensive.

Upon analyzing the IEEE 802.11 protocol and the data transfers occurring during a streamed content, we come easily to the conclusion that the protocol is not optimized for constant, periodic transfers as in streaming, but for sporadic, smaller transfers. Therefore, the built-in power management of 802.11 does not succeed in taking full advantage of the idle periods during the transfer.

Various research has been done over recent years toward minimizing power consumption for the network interface. Some have looked at improving upon the default IEEE 802.11 power management built-in, others have proposed new power policies that reduce the time the interface must be in active mode, and place it in a low power the rest of the time. But these techniques, while achieving important savings, trade off some quality for lower power in most of the cases, because a prediction model is used to forecast future network transfers.

We take a different approach: instead of predicting, we preprocess each multimedia stream beforehand and annotate the needed information to the stream. The annotations are specific to each stream, but they are general enough for all the clients to be able to take advantage of them. Since we are using data gathered through pre-processing, we can achieve a level of accuracy much higher than other techniques based on a prediction model using the recent history. Our technique has both server and client side components. The server pre-processes the data and annotates it, while the client extracts the annotated information and performs optimizations based on it at runtime.

Patterns in the stream are mainly due to the quasi regular nature of multimedia at lower levels, because their encoding/decoding is composed of processing filters, applied in a specific order. This regular behavior is confirmed by recent research [13]. Changes are only introduced by variations in the input data [11] and the algorithm itself.

At higher levels (network transfer), multimedia streams are packetized in a regular, hierarchical way (e.g. MPEG system layer), with predictable parsing and consumption by the client players. In conclusion, knowledge of data characteristics can be exploited at all levels in a multimedia streaming application, and is especially important for portable devices

where battery life and thus mobility are of utmost importance.

Annotations prove beneficial twice: first, because they can be done “off-line” (statically) while saving the mobile device from the burden of analyzing the data at runtime; and second, because annotating preprocessed data is more accurate than predicting its behavior.

The report is organized as follows: we start with an overview of data annotation in general and for multimedia in special. Next, we present some related work and show how our approach is different, complementary or can benefit from it. Then, we introduce our approach for applying annotations to wireless network power management and we describe the technique for managing the network card and the intermediary buffer. Our experimental setup and results are presented next. The last section concludes the paper and discusses future work.

2 Data annotation

Annotations have previously been applied to other domains. In compilers, annotations are sometimes used for retaining as much from the original semantics as possible, in an automated way. At the same time, a programmer can manually annotate the source code to pass information or hints to the compiler [9]. For example, register assignment for variable in C falls in this category. Other examples include the use of ‘pragma’ directives (C, C++, Ada).

In a same way, the process of annotating the data stream is either automated (performed statically, by an analysis step) or under user supervision (here the user can, for example, specify which parts or objects of the video stream are more important and should be best preserved in a power-quality trade-off).

Data annotation is the process of analyzing the content of a stream of data and adding the collected information to the stream itself; the annotations will later be used for run-time content-aware optimizations. Annotations typically capture patterns or trends in the data stream that are difficult/impossible or too time-consuming to gather at run-time on the mobile device and that can be later exploited for either power or performance benefits.

The annotations can be performed either statically (for example off-line annotation/profiling in case of on-demand media serving, where the information is preprocessed and stored at the media servers) or dynamically (in case of live streams, the annotation can be done on-the-fly by an intermediary proxy node).

Annotations can be used at either the client side, for an increased user experience, or at a proxy node, which may perform additional on-the-fly operations on the data stream to adapt it to the capabilities of the client (transcoding, etc.). These annotations may prove useful in estimating the required bandwidth for communication, estimating computation or applying more aggressive QoS trade-offs based on image content.

Multimedia applications are typically studied at different abstraction layers: application, middleware/network, OS, hardware. Each of these layers can use, and benefit from, additional information on the data stream. A classification of all possible applications of annotations is presented in Figure 1.

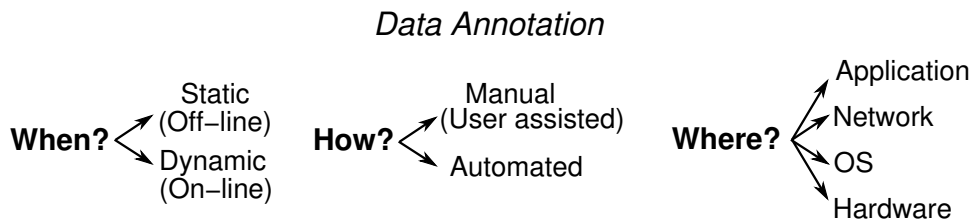


Figure 1: Classification of possible annotations

Specifically, in this paper we focus our attention at the network (wireless card) level. We assume that the annotation is performed off-line, at the server end (by profiling an extensive data set, representative for the application domain). We focus on multimedia streaming of content, in particular MPEG encoded video streams.

The advantage of annotating the data off-line is two-fold. First, there is no overhead for doing all the work at runtime, by the client device. Second, because the information is known in advance, more optimizations are possible (for example the network optimizations).

3 System architecture

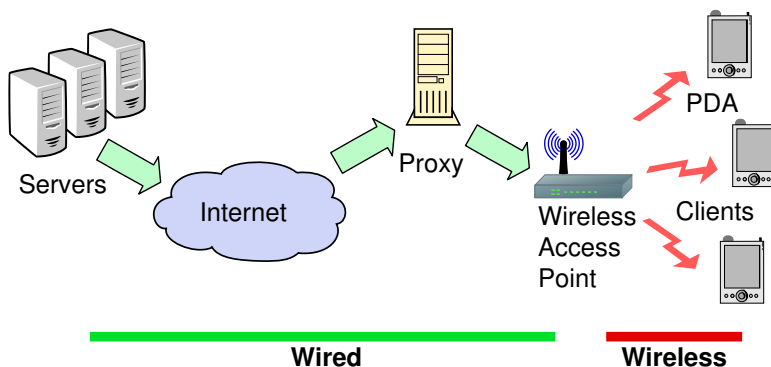


Figure 2: System framework

We assume the system model depicted in Figure 2. The system entities include a multimedia server, an optional proxy node that can perform various optimizations on the stream (e.g. transcoding), the users with low-power wireless devices and other network equipment along the way. The multimedia servers store media content and stream videos to clients upon requests issued by the users on their mobile devices.

The communication between the client device and the servers can be routed through an optional proxy server – a high-end machine that has the ability to process the video stream in real-time. The proxy node can also perform in-line profiling/annotation for real-time video streaming (live applications like videoconferencing are examples).

Annotations can be generated and added to the video stream at either the server side or the proxy node, therefore saving the client of any additional work.

4 Related work

In this section we are reviewing existing research work that focuses on analyzing the input data stream and deriving various techniques for improving either communication or computation in streaming applications.

The Mesdat research group studies various data-shaping algorithms for mobile multimedia communication. They profile and annotate images for improving transmission over a wireless channel (bandwidth, latency). In [8] the image is compressed according to dynamic conditions and requirements. Content adaptation is classified based on time (static, dynamic), content (for best compression) and goals of technique (constrained bandwidth, display size, response time).

Chandra performs an informed quality-aware transcoding in [5], based on image characteristics. He finds that a change in JPEG quality factor (compression metric controlled by quantization steps) directly corresponds to information quality lost. A prediction for computational overhead is applied, which approximates number of basic computation blocks based on image size, color depth and can predict output size for a particular transcoding.

In [6], the authors analyze the characteristics of images from web sites (distribution of gif or jpg images, size, colors, quality). They classify images in: bullets, lines, icons, banners, true images based on heuristics and analyze various transcoding techniques for reducing image size (reducing spatial geometry, reducing the number of unique colors, changing the image format or compression).

The work presented so far is different from our approach in that it only handles static images, mainly from web pages. We are instead focusing our efforts on continuously changing content (video streaming).

Tripathi and Claypool study different ways to reduce bandwidth in network transmission in [14], by either temporal scaling (dropping frames), quality scaling (reducing quality of frames) or spatial scaling (changing the size of frames). The quality degradation is evaluated through a user study. In our technique we are not altering the content, we are simply adding annotations to the stream.

Traffic reshaping was studied by Chandra et al in [7] and [3], for typical Microsoft media, Real media and Quicktime streaming formats. They show that the default power management in IEEE 802.11 is limited and propose a policy for shaping the network traffic to allow for longer idle periods between transfers. By placing the wireless card in power saving mode during idle times, they are able to achieve high energy savings for the network interface. However, the history based approach means that prediction of transfer times is not perfect and packet dropping occurs (depending on the format). This forces the transmission to reduce the quality (bitrate) of the video stream. The authors conclude that some formats are harder to predict at the network level without understanding the packet semantics. Further

work by Chandra in [4] further studies different traffic shaping mechanisms and discusses their efficiency.

As an improvement to the previous history-based prediction, [15], Wei et al propose a statistical prediction scheme for future idle intervals and show that the results achieved are better than simple averaging. However, the drop rate reaches higher levels as the higher energy savings increase (up to 50% drop rate).

Our work closely resembles the traffic reshaping work above and improves over the algorithm used: instead of relying on a prediction scheme, our technique employs a preprocessing phase in which we annotate the stream with the actual consumption rate of the video player. As a streaming protocol, we chose to use a pull data mechanism (e.g. http) instead of push data (real-time protocol), because we find that an increased number of video streams is offered this way on the Internet. Our conclusion is that the application of our technique can prove beneficial to traffic shaping by eliminating the packet loss associated with wrong predictions.

In a different way of using the behavior of programs, Anand et al propose in [1] a self-tuning power management (STPM) technique that adapts its behavior to access patterns of the applications and characteristics of the network interface. They show important power savings compared with the default IEEE 802.11 power management and reduced delays. They show how application hints can bring additional benefits. Applications studied include the NFS file system transfers, audio streaming and remote X clients.

Application characterization for wireless power management is also studied in [16] where the authors argue that power reduction and quality degradation depend on the application and the user. They present an approach for identifying the class of running applications, dynamically adapting the power saving policy to the detected profile and applying the corresponding power/performance trade-offs.

The work above is different from ours in that it focuses at adjusting network parameters to changes in the applications running rather than in the input data from a single application.

Energy management for buffers used in streaming applications is studied by Cai et al in [2], for a generic producer/consumer setup, for fixed production and consumption rate and generalized for multiple or variable rates, intermittent consumption and multiple consumers. Optimal buffer size computation formulas are presented in each of these cases, to reduce energy consumption. For our work, buffer memory power is not significant, because enough memory is easily available in a handheld device or laptop, but it can prove important for an embedded device specifically designed for multimedia streaming (e.g. phone), where memory is a more scarce resource.

In summary, compared with the afore mentioned related work, our technique works on dynamic content (video), performs most of the work (profiling and annotation) off-line and this allows us to more accurately estimate transfer rates and buffer network packets with a minimal overhead at runtime.

5 Approach

Our technique is based on annotations that are generated during a preprocessing step, at the server side. The annotations are embedded into the MPEG stream and sent over the network. The stream flows through the network nodes and reaches the end point of the wired part, the wireless access point. Here network packets combined into burst transmissions are sent the wireless link to the mobile devices. The wireless part of our framework is explained in more detail in Subsection 5.2.

At the client side, data received from the network is fed over to the decoder. To accommodate the burst nature of the transmission with the continuous, periodic data consumption by the decoder, an intermediary buffer is placed in between (Subsection 5.3). The decoder (final step) parses the MPEG stream (Subsection 5.4), decodes the audio and video channels and sends the decoded data to the corresponding output interface. The next subsections explain each of these components.

5.1 Preprocessing and Annotation

At the server side, each video stream is first passed through an instrumented video decoder (player), which records information about bandwidth variation over time. The relative size of MPEG packets and the time stamps of the requests are used to build a bandwidth profile for each video clip and generate annotations. Each annotation specifying data size and time of packets is added to the stream. This information is used at the wireless access point when the network packets are grouped into bursts and at the client when the burst data is requested and buffered before being sent to the decoder.

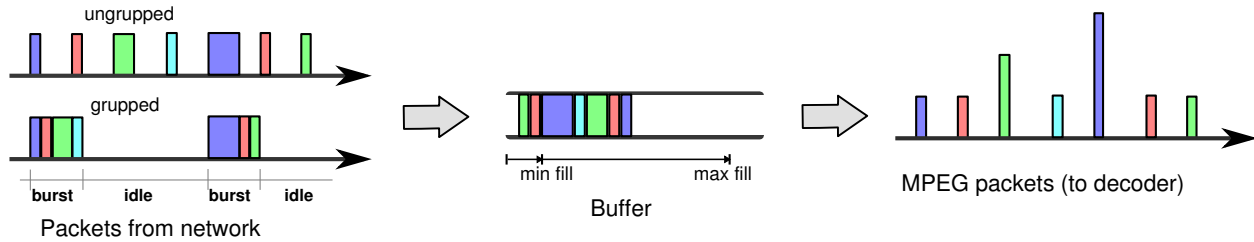


Figure 3: Wireless burst transmission and buffer management

5.2 Wireless Interface

The wireless interface is one of the vital components in a mobile device, also accounting for a large part of its energy consumption. Wireless networking is governed by the IEEE 802.11 standard[12], with a number of versions (a, b, g) which differ in bandwidths and channel frequencies. Wireless technology was originally intended for irregular, sporadic data transfer and therefore its default power saving mode is optimized for those patterns. Newer application, like multimedia streaming on the other hand, tend to use the wireless interface

at all times, continuously transferring data, thereby minimizing the opportunities for power management.

The default power management in IEEE 802.11 (PSM) employs a rendezvous approach in which the client and the access point cooperate. The access point periodically sends out beacons (every 100ms by default), which the client monitors for indication of available packets for delivery. Between these beacons, the client adapter can go into low power (idle or sleep) mode. -This is in contrast to the continuous access mode (CAM) when the client is always in an active (idle) state, ready to process requests.

In our approach we follow a traffic shaping mechanism similar to the one described by Surendar in [7] and subsequent work. The traffic is buffered into bursts of larger size and transmitted to the client periodically. Between the burst transmissions, the network card is switched into low power sleep mode. The card is switched back to active mode just before the next transfer (more details about the network parameters are presented in the experimental section). Our work is similar in spirit to the one mentioned above but we are employing an annotation based technique that, unlike prediction, allows us to avoid the high packet drops when prediction proves wrong. The annotation works by profiling the media content off-line and sending the information about client data consumption (mainly dependent on the MPEG stream structure) along with the media.

5.3 Buffer Management

Because the data is transferred in bursts over the wireless link, a buffer mechanism is required to allow for a smooth playback at the client side. We insert a custom buffer between the network layer and the application, which fills up with streamed content during a burst transmission and is consumed by the player during playback.

The buffer is periodically refilled with data through burst transfers, using a simple algorithm:

```
if buffer_size < min then  
  while buffer_size < max do  
    request up to (max - buffer_size) bytes from server  
  end while  
end if
```

The size of the buffer should be large enough to allow larger burst transfers and longer idle periods, but smaller enough to minimize the delay for the video player. Based on experimentation, we chose a buffer size of 64Kb for our video clips, which is the minimum size that reduces packet drops to zero and allows for important energy savings (Figure 4), while at the same time keeping the introduced delay to under one second (reasonable for on demand media). For buffers larger than 64Kb there is virtually no more gain in either savings or packet loss, but the delay increases. For smaller buffers, most videos start experiencing frame drops, because of limited buffer capacity.

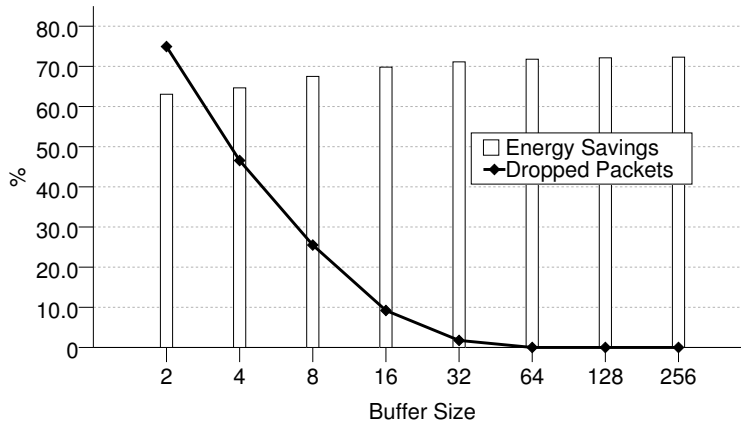


Figure 4: Buffer size tradeoff

The flow of the stream from the network to the decoder is presented in Figure 3. The left part of the figure depicts the wireless part of the framework. Above, the network packets arrive in their original order, with no grouping. Packet size varies depending on their payload. The picture below it shows the same packets after our burst buffering is employed. Packets are bundled together in bursts such that they do not overflow the size of the intermediary buffer, based on the annotations which carry their size and time distribution. From the wireless interface, the packets are stored in the buffer, show in the center (with its min and max thresholds). We chose to use thresholds (and not just run the buffer to completion and completely fill it up) in order to avoid interruption in the flow of data to decoder and to avoid a buffer overflow situation (if a burst transmission arrives and there is still unused data in the buffer). The packets are read from the buffer one by one, in their original sequence and in a transparent way by the MPEG decoder, shown on the right side. The decoder processes the stream in a normal fashion, without knowledge of the burst transmission taking place.

5.4 MPEG Stream Format

The MPEG stream format is organized in a hierarchical way. Each audio, video or data source is compressed and packetized in a PES (Packetized Elementary Stream). For example, a MPEG video compressed stream is organized top-down into sequences, groups of pictures (GOP), pictures, slices, macroblocks and blocks, each level grouping a number of object at the next level. The packetized elementary streams are then split into payloads packets and combined into MPEG system streams, which contain in addition other header, padding and special purpose fields. The PES streams can be multiplexed into either a program or a transport stream, which differ by the size of packets, fixed or variable, number of programs per stream and ease of recovery and demultiplexing.

The conclusion is that in each of the formats, there is a very clear hierarchical organization (Figure 5). This implies that a video player has to parse the MPEG stream before being able

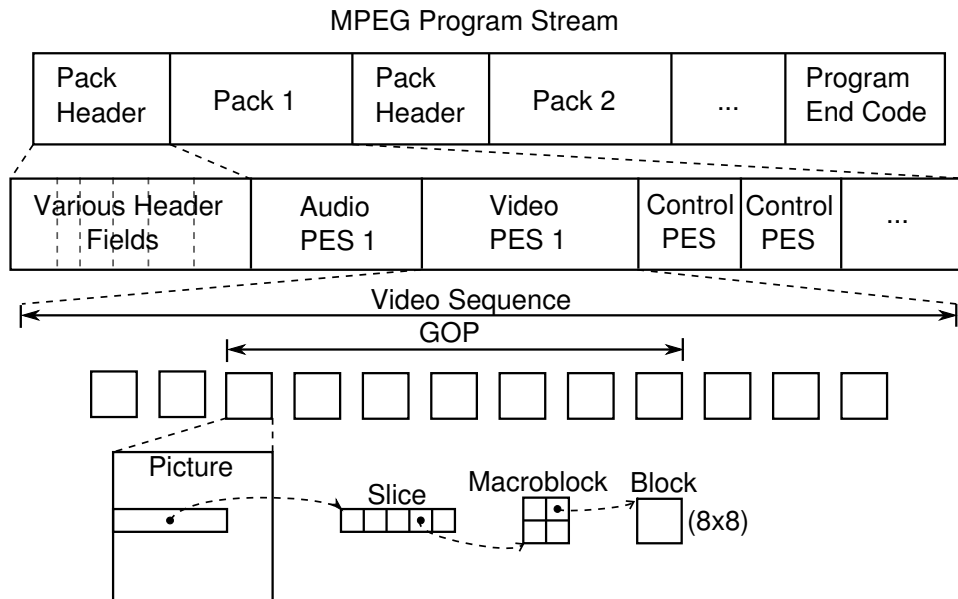


Figure 5: Hierarchical MPEG stream organization

to send the data to the decoders (video and audio). Because the MPEG organization is fixed and determined at encoding time, all the players are parsing and consuming the data in a similar way, based on its organization into different layers. We are analyzing the consumption pattern as a pre-processing step and use the information collected to group and transfer the packets that are likely to be consumed together in the same burst.

This information is obtained by instrumenting a video player and tracking the data reads from the MPEG parser module, which is then annotated to the stream itself. Annotations store information about the exact data size and time when it will be required by the decoder, ahead of time. Since the annotations are relatively small compared to the data (on the order of bytes for each tens of kilobytes of data), they don't have a significant impact on the transferred data size. The annotations can be stored either inside the MPEG stream (unused header locations, padding packets) or sent on a special, dedicated control channel.

6 Experimental Results

6.1 Experimental Setup

For playing the videos, we chose the MPEG video player, which is part of Berkeley MPEG Tools, because of code availability and ease of use. We instrumented the video source to capture the size and time corresponding to all packet reads from the incoming stream (during the preprocessing phase). The information is then annotated to the stream and used during run-time to drive the burst transmission (grouping of packets into bursts and buffer

management).

Each video clip was profiled in this fashion, an example graph showing the data read (total, average, frame size) is shown in Figure 6. From this profiling we collect the packet consumption rates (size, time) and create the annotations. We observed that the majority of the packets are distributed in multiples of 2Kb and the arrival time is on the average a multiple of 40ms. Since the arrival time of the packets is less than the IEEE 802.11 beacon interval of 100ms, the default power management of the wireless interface fails to put the card into low power mode most of the time. The savings achieved by the default power management are minimal for multimedia streaming in this case. We present our results relative to the savings achieved by the default PSM power management present in the IEEE 802.11 protocol.

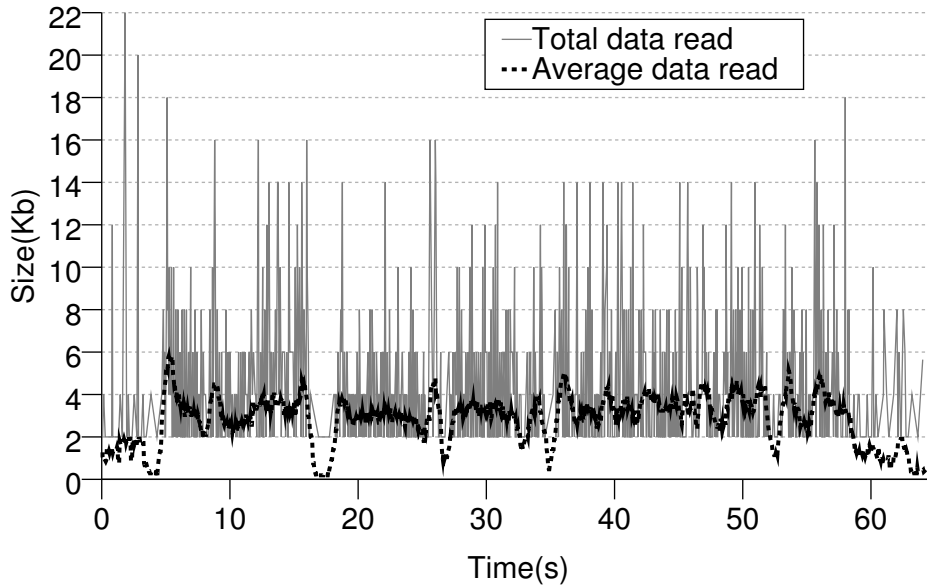


Figure 6: Example profiled video clip (catwoman)

As input sources, we chose a large number of video clips formatted for a typical PDA (video sizes around 320x200), downloaded from <http://pocketmovies.net>. The clips are MPEG encoded and have bitrates between 38Kb/s and 82Kb/s, with running times varying from 1 to 12 minutes. Some clips are action types (movie trailers), while others are animations, commercials or short films. We selected the video clips to cover the possible types of video with respect to bandwidth consumption patterns (from mostly still to very dynamic scenes).

The network card we simulate (a 2.4 GHz Wavelan card [10]) has four power states: transmit data, receive data, idle and sleep. Power numbers are taken from the above mentioned reference: 1.67 W transmit data, 1.43 W receive data, 1.32 W idle, 0.18 W sleep mode. Transitioning from sleep to active mode takes 250 μ seconds, but we are waking up the network

Table 1: Total energy savings/package drop for different video clips and buffer sizes

Video Clip	Energy Savings (%)				Package Drop (%)			
	16 Kb	32 Kb	64 Kb	128 Kb	16 Kb	32 Kb	64 Kb	128 Kb
catwoman	60.01	62.24	63.27	63.79	1.80	0.00	0.00	0.00
405themovie	68.95	70.27	70.96	71.31	4.32	0.50	0.00	0.00
blockbuster	66.20	67.77	68.55	68.95	0.00	0.00	0.00	0.00
ep2_clone	70.72	72.01	72.60	72.92	0.00	0.00	0.00	0.00
episodeIII	63.83	65.61	66.47	66.93	1.38	0.00	0.00	0.00
getinspired	62.50	64.38	65.34	65.82	4.49	0.00	0.00	0.00
grimm	63.50	65.29	66.20	66.65	0.87	0.00	0.00	0.00
hellweek	67.21	68.75	69.50	69.88	1.59	0.00	0.00	0.00
gobletoffire	66.13	67.72	68.51	68.93	1.47	0.00	0.00	0.00
hunter	71.43	72.62	73.20	73.50	0.00	0.00	0.00	0.00
iceage2	60.68	62.68	63.68	64.18	2.67	0.00	0.00	0.00
ice_age	60.19	62.13	63.25	63.78	5.66	0.00	0.00	0.00
i_robot	54.80	57.28	58.60	59.21	8.75	0.00	0.00	0.00
king_kong	58.64	60.80	61.91	62.45	2.00	0.29	0.00	0.00
meeting_agnus	72.56	73.73	74.28	74.56	0.28	0.00	0.00	0.00
returnoftheking	65.13	66.85	67.67	68.10	1.11	0.00	0.00	0.00
sallyangela	58.31	60.82	61.92	62.46	0.14	0.00	0.00	0.00
saturday	66.84	68.36	69.11	69.51	1.24	0.00	0.00	0.00
shrek2	55.16	57.44	58.76	59.41	22.99	0.57	0.00	0.00
spaceodddity	69.81	71.12	71.78	72.13	9.21	1.76	0.00	0.00
spiderman2	54.49	57.00	58.32	58.95	10.01	0.00	0.00	0.00
incredibles	56.22	58.61	59.84	60.44	9.55	0.00	0.00	0.00
timescape	71.87	73.07	73.63	73.92	0.61	0.00	0.00	0.00
underground	70.11	71.33	71.96	72.29	0.00	0.00	0.00	0.00
wronglanding	68.91	70.37	71.07	71.43	1.37	0.00	0.00	0.00
zeroonezero	60.17	62.24	63.32	63.83	4.87	0.00	0.00	0.00

card 10ms ahead of them to account for any other system delays (clock resolution).

We simulated a wireless network that provides an average bandwidth of 2Mbps, which is a reasonable value for a 11Mbps wireless network. The assumption is that the access point intelligently schedules each client for transfers, and sends the data in a burst transmission for each client (the burst itself might have been prepared by the proxy node or the server itself). This is needed because network errors or fragmentation would decrease the efficiency of burst transmission. The effect of network noise due to cross-traffic between other nodes is not considered and is subject of future work. If the network noise is too high, burst transfers could potentially be affected, which would reduce the efficiency of our technique (lower power savings).

Content from the network passes through the intermediary buffer (placed between the wireless network interface and the decoder), which stores data transferred during a burst transmission and serves it to the video decoder as requested. Based on experimentation we chose a buffer size of 64Kb, in order to minimize the delay and avoid frame loses for our selection of video clips. For our input streams, the delay introduced is of the order of one second, acceptable for a one way transmission of content.

6.2 Results

The results from our experiments are presented in Table 1. For the video clips selected, energy savings of 60 to 75% are possible with no image quality degradation. We show the results for buffer sizes from 16Kb to 128Kb, but in practice we noticed that there are virtually no increase in savings for values larger than 64Kb. Some video clips experience packet drops for a 32Kb buffer size and even more for a 16Kb, due to spikes in their packet sizes (due to rapid scene movements or sudden scene changes inside the content). The video clips that have smaller packets in their streams perform better, even for smaller buffer sizes.

Compared with results previously reported by Surendar et al in [3], our technique yields similar savings (showing that a burst transmission is both feasible and desirable multimedia streaming), but without the high drops that occur if prediction is used (up to 50% in the worst case), and without the need to decrease the quality (bitrate) of the video stream. Our approach is not susceptible to these packets drops due to prediction errors. Performance degrades in both approaches in the presence of high network errors (which of course would degrade the original video transmission too).

7 Conclusion

Annotations prove useful in improving power management of network wireless devices, for streaming multimedia. We have shown an approach to shaping network traffic into bursts of data, based on annotations, while turning the network card in a lower power mode in between the bursts. This approach leads to up to 75% energy savings for the network card, with practically no quality degradation (frame loss).

We compared our approach to previous and related work and showed how our technique is either better or complementary to them.

As future work, we plan to extend our annotation driven techniques to the other possible areas mentioned in the beginning of the paper (hardware, application) and to investigate dynamic applications of annotation in addition to the static one.

References

- [1] M. Anand, E. Nightingale, and J. Flinn. Self-tuning wireless network power management. *Wireless Networks*, 11(4):451–469, 2005.
- [2] L. Cai and Y. Lu. Energy management using buffer memory for streaming data. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(2):141–152, 2005.
- [3] S. Chandra. Wireless network interface energy consumption. *Multimedia Systems*, 9(2):185–201, 2003.
- [4] S. Chandra. Energy conservation in ad hoc multimedia networks using traffic-shaping mechanisms. *SPIE*, 5305:40, 2004.
- [5] S. Chandra and C. S. Ellis. JPEG compression metric as a quality-aware image transcoding. In *USENIX Symposium on Internet Technologies and Systems*, 1999.
- [6] S. Chandra, A. Gehani, C. S. Ellis, and A. Vahdat. Transcoding characteristics of web images. In M. Kienzle and W. chi Feng, editors, *Multimedia Computing and Networking (MMCN'01)*, volume 4312, pages 135–149, San Jose, CA, jan 2001. SPIE - The International Society of Optical Engineering.
- [7] S. Chandra and A. Vahdat. Application-specific network management for energy-aware streaming of popular multimedia formats. *Usenix Annual Technical Conference*, 2002.
- [8] D.G.Lee, D.Panigrahi, and S.Dey. Network-aware image data shaping for low-latency and energy-efficient data services over the Palm wireless network. In *World Wireless Congress (3G Wireless)*, 2003.
- [9] S. Z. Guyer and C. Lin. An annotation language for optimizing software libraries. In *Domain-Specific Languages*, pages 39–52, 1999.
- [10] P. Havinga. Mobile multimedia systems. *University of Twente, Ph. D Thesis*, pages 90–365, 2000.
- [11] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan. Variability in the execution of multimedia applications and implications for architecture. In *International Conference on Computer Architecture*, pages 254–265, 2001.

- [12] Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Standard 802.11, June 1999.
- [13] T. Sherwood, S. Sair, and B. Calder. Phase tracking and prediction. In *International Symposium on Computer Architecture*, 2003.
- [14] A. Tripathi and M. Claypool. Improving multimedia streaming with content-aware video scaling. Technical Report WPI-CS-TR-01-02, Worcester Polytechnic Institute, 2001.
- [15] Y. Wei, S. Chandra, and S. Bhandarkar. A statistical prediction-based scheme for energy-aware multimedia data streaming. *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, 4:2053–2057, 2004.
- [16] A. Weissel, M. Faerber, and F. Bellosa. Application characterization for wireless network power management. *Proceedings of the International Conference on Architecture of Computing Systems (ARCS2004)*, 2004.