Low Power Video Streaming for PDAs

Morihiko Tamai, Keiichi Yasumoto, Naoki Shibata and Minoru Ito
Graduate School of Information Science at Nara Institute
of Science and Technology, Ikoma, Nara 630-0192, Japan
{morihi-t,yasumoto,n-sibata,ito}@is.aist-nara.ac.jp

Abstract

In this paper, we propose an battery-aware video streaming system which can extend battery life of portable computing devices to a desired playback duration in streaming video playback. The proposed system transcodes a video stream from the content server to the lightweight stream to save power consumption. Our system includes another power saving mechanism where video data is received and stored in a buffer on the portable device at the maximum transmission speed, and the wireless NIC is powered off until the buffer becomes empty. We define a power consumption model on the portable device to determine several video parameters satisfying a user requirement. We have implemented the system and carried out several experiments. As a result, we have confirmed that our system can extend the battery life up to 2.5 times and that the predicted battery life can be estimated accuratery using our model.

1. Introduction

Recent performance improvement of portable computing devices (e.g., PDAs and cellular phones) is remarkable and they have now wireless communication capabilities. Moreover, wireless LANs are rapidly becoming popular and available in many places as "hot spots." These situations encourage us to see video contents such as movies and live sportscasts everywhere and whenever we want. However, portable devices consume a lot of battery amount for video playback in real time streaming due to wireless communication and decoding of compressed video data. So, it is difficult to play back a video for a long duration.

As a typical situation, let us suppose that a user is suddenly given one hour free time and he/she want to play back a short movie (within one hour, say, 57 min.) with his/her portable device by receiving the video stream via a wireless LAN. However, the device's battery may not last for one hour while playing back the video. For such situations, in this paper, we propose a low power video streaming method which applies some power control techniques in order to extend the device's battery life to last for the specified playback duration.

Most of existing studies on power saving have focused on low-power hardware design [1, 2, 3]. On the other hand, there are some software based approaches to power saving. In [4], power saving is achieved by a power control method based on channel condition of wireless LANs. In [5], a power saving method which selectively discards video frames (I, P or B frames) to decrease the number of bits transmitted over the wireless link is proposed.

In this paper, we propose two methods to extend battery life in video streaming for portable devices. In the first method, we save power consumed for video decoding and displaying using a transcoding technique [6] which converts the original video stream to a lightweight stream (means that the receiver can play it back for a longer duration than the original one). For the purpose, we introduce an intermediate node, called *transcoding proxy*, between a server to a client to relay video streams from servers to clients after transcoding the video streams to lightweight streams in real time. In the second method, we save power consumed for wireless communication by periodic bulk transfer of a video data and by supplying power to the wireless network interface card (called WNIC, hereafter) only during data transfer, while playing back the video stored in a buffer.

We have implemented the system and carried out several experiments. As a result, we have confirmed that our system can extend battery life up to 2.5 times and that using our algorithm, we can estimate the battery life within 9% accuracy.

2. Low Power Video Streaming Method for PDAs

2.1. A Strategy for Power Saving

The power consumed by a portable device while playing back a video stream delivered from a server consists of (1)

Power consumed by OS, (2) Power consumed by the backlight, (3) Power consumed for the WNIC and for processing packets, and (4) Power consumed for decoding and drawing video frames. In our method, we focus on saving power for the (3) and (4).

2.1.1 Power Saving Technique for Video Decoding and Drawing

The power consumed for decoding and drawing video frames is proportional to the product of the frame rate (i.e., the number of frames drawn per second), the picture size (i.e., the number of pixels), and the bit rate of the video. Therefore, it is possible to save the power by reducing the values of these parameters.

For such power saving, one of simple solutions is to prepare several files with the different parameters for each video in a server so that a user can select one depending on his/her portable device's battery amount. However, this approach is costly for content providers or may not control the playback duration minutely when only small number of files are prepared. Therefore, we have adopted another approach to use the transcoding technique where an intermediate proxy node (called the *transcoding proxy*) receives a video stream from a server, converts it in real time into a lightweight stream, and forwards it to the user's portable device.

2.1.2 Power Saving Technique for WNIC

We have carried out some preliminary experiments to investigate the power consumed by the WNIC (an IEEE 802.11b LAN card is used) of the portable device. Then, we have confirmed that the WNIC consumes much power while it is turning on, and that the power consumed does not vary so much depending on the data transfer rate. When we let a portable device play back a video stream from a remote server, usually the server transmits the stream at the same transmission rate as the originally encoded bit rate of the video. In this case, the power must be supplied to the WNIC all the time during playback. When the available bandwidth of the wireless network is higher than the bit rate of the video, we can download a portion of the video stream at the maximum transmission rate in shorter time than the playback duration of the portion. So, by using such bulk transfer and by storing the downloaded portion in a local buffer of the portable device, we can stop supplying the power to the WNIC for the time difference in order to save power. The detailed procedure to save power at a portable device based on the above consideration is as follows:

(1) allocate a buffer with the size of M bits.
(2) turn on the WNIC, invoke two threads T_r and T_p in parallel where T_r receives the video stream at B_{bf} bps (here, we assume that B_{bf} is greater than the original bit rate b_1 of the video) and stores the data in the buffer, and T_p plays back the video from the buffer.
(3) when the buffer is filled with the stream data, suspend thread T_r and turn off the WNIC.
(4) when the buffer becomes nearly empty, turn on the WNIC and resume thread T_r.
(5) repeat the steps from (3).

Hereafter, we call the above playback mechanism as the *buffered playback*.

The buffered playback can be used for "pull" type data sources, that is, video files stored in a server which can be downloaded at the arbitrary transmission rate, for example, by HTTP. However, it is not straightforward to apply the buffered playback to "push" type data sources since in such a data source, a video is transmitted in real-time as a stream whose transmission rate is the same as the original bit rate b_1 of the video. So, we introduce another intermediate node called the *buffering proxy* which is located between the transcoding proxy (or the content server) and the portable device. The buffering proxy receives the video stream from the transcoding proxy (or the server) and stores the data in its buffer (the size of the buffer is the same as the portable device). When the buffer is filled, the portable device downloads the data from the proxy at the maximum transmission rate B_{bf}. Note that the start time of the video playback at the portable device will be delayed for M/b_1 sec. We call the delay as the *buffering delay*.

In general, the transcoding proxy and the buffering proxy can be executed on different nodes. Here, for the sake of simplicity, we assume that those proxies are executed on a single node.

2.2. Organization of the Low Power Video Streaming System

We show the organization of the proposed low power video streaming system in Fig.1.

Each user sends to the transcoding proxy a video delivery request including the location of the video, the desirable playback duration, the preference for video quality, and the terminal information. From the above information, the transcoding proxy converts the video stream received from the content server to the lighter stream with the appropriate video parameters, and transmits the converted stream to the portable device.

We call the following parameters as the *battery-aware parameters*: (1) the frame rate, (2) picture size, and (3) the bit rate of a video and (4) the buffer size for the buffering playback. When the remaining battery amount of the portable device is sufficient for the playback of a video with

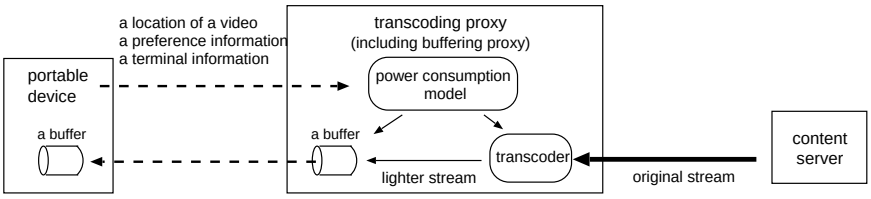

Figure 1. Organization of the proposed video streaming system

the lowest quality, there may be multiple solutions consisting of different battery-aware parameter values. On the other hand, the battery amount is not sufficient, there may be no solution. In the proposed method, one combination of those parameter values is selected based on the user's preference.

3. Algorithm to Decide Battery-aware Parameter Values for Desired Playback Duration

In this section, we define a power consumption model consisting of battery-aware parameters for the portable device. Based on the model, we compute the expected battery life for each combination of parameter values.

3.1. Power Consumption Model

Our power consumption model assumes that the following conditions are satisfied as long as a movie can be played back without frame skipping.

1 Total electric power consumed by a portable device is the sum of $S_0, V(r, f, b)$ and $N(B)$, where S_0 denotes the sum of the power consumed by backlight and the operating system, $V(r, f, b)$ denotes the power consumed by decoding and displaying a movie whose picture size (the number of pixels), frame rate, and bit rate are r, f and b, respectively, and $N(B)$ denotes the power consumed by the WNIC, when the average transmission rate is B.

2 S_0 is a constant.

3 $V(r, f, b)$ is proportional to the product of r, f and b. That is, with a constant c_V,

$$V(r, f, b) = c_V(rfb) \tag{1}$$

holds.

4 $N(B)$ is linearly increasing with B. Hence, with constants c_N and N_{base},

$$N(B) = c_N B + N_{base} \tag{2}$$

holds.

5 The WNIC consumes N_{base} of the electric power for the time during suspending (the time taken to go into the suspension mode from the normal mode) and resuming (the time taken to go back to the normal mode from the suspension mode).

6 The electric power consumed by the local storage device (e.g., flash memory) is negligible.

The fully charged battery amount of a portable device is denoted by E_0. Let r_0, f_0 and b_0 be the number of pixels in a frame, the frame rate and the bit rate of video v_0, respectively. We assume that the portable device can playback v_0 without frame skipping.

Hereafter, the time until a portable device is switched off due to battery exhaustion, is said to be the *battery life*.

Let T_{SVN} be the battery life while we played back v_0 by streaming on a portable device with battery amount E_0. Let V_0 be the power consumed by decoding and displaying v_0. Let N_0 be the power consumed by wireless device when receiving data of v_0. Then, we have

$$E_0 = T_{SVN}(S_0 + V_0 + N_0) \tag{3}$$

Let T_{SV} be the battery life while playing back v_0 from the local storage device with battery amount E_0. Then, we have

$$E_0 = T_{SV}(S_0 + V_0) \tag{4}$$

Let T_{SN} be the battery life while only receiving v_0 by streaming (i.e., do not playback) with battery amount E_0. Then, we have

$$E_0 = T_{SN}(S_0 + N_0) \tag{5}$$

Let T_{base} be the battery life while the WNIC is turned on and the portable device performs neither receiving, decoding nor displaying with battery amount E_0. Then, we have

$$E_0 = T_{base}(S_0 + N_{base}) \tag{6}$$

Let T_S be the battery life while the WNIC is turned off and the portable device performs nothing. Then, we have

$$E_0 = T_S S_0 \tag{7}$$

From Eqs. 5, 6, and 7, and by using $X_0 = N_0/N_{base}$, we get

$$X_0 = \frac{T_{base}(T_S - T_{SN})}{T_{SN}(T_S - T_{base})}$$

Also, from Eq. 2, we get $X_0 = (c_N b_0 + N_{base})/N_{base}$. Rewriting it with $\alpha = b_0/(X_0 - 1)$, we get $N_{base} = \alpha c_N$. Assigning the above equation to Eq. 2, we get $N(B) = (\alpha + B)\, c_N$.

Since

$$\frac{N_0}{N(B)} = \frac{\alpha + b_0}{\alpha + B}$$

holds, rewriting it with $\beta(B) = (\alpha + b_0)/(\alpha + B)$, we get

$$N(B) = \frac{N_0}{\beta(B)} \tag{8}$$

The actual values of the above parameters $T_{SVN}, T_{SV}, T_{SN}, T_{base}$ and T_S can be measured for each combination of a portable device and a WNIC. With those values, we propose a method to calculate the expected battery life.

Expected battery life when only transcoding is used

Let r_1, f_1 and b_1 be the parameters of movie v_1. Let V_1 be the power consumed by decoding and displaying v_1. The expected battery life T_{tr} of v_1 can be computed from the following formula.

$$E_0 = T_{tr}(S_0 + V_1 + \frac{N_0}{\beta(b_1)}) \tag{9}$$

Since $V_0 = c(r_0 f_0 b_0)$ and $V_1 = c(r_1 f_1 b_1)$, using

$$m = \frac{r_0 f_0 b_0}{r_1 f_1 b_1} \tag{10}$$

Eq. 9 is transformed to

$$E_0 = T_{tr}(S_0 + \frac{V_0}{m} + \frac{N_0}{\beta(b_1)}) \tag{11}$$

Introducing equation including T_m,

$$E_0 = T_m(S_0 + \frac{V_0}{m} + N_0) \tag{12}$$

and from Eqs. 3 and 5, we get

$$T_m = \frac{T_{SVN} T_{SN}\, m}{T_{SVN}(m-1) + T_{SN}} \tag{13}$$

Introducing equation including T_β

$$E_0 = T_\beta(S_0 + V_0 + \frac{N_0}{\beta(b_1)}) \tag{14}$$

and from Eqs. 3 and 4, we get

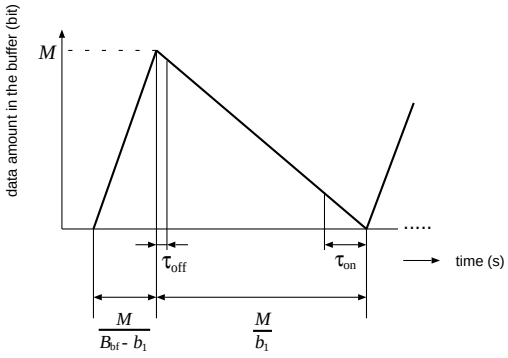

Figure 2. Variation of the data amount in the buffer during buffered playback

$$T_\beta = \frac{T_{SVN} T_{SV} \beta(b_1)}{T_{SVN}(\beta(b_1) - 1) + T_{SV}} \tag{15}$$

Hence, from Eqs. 3, 12 and 14, we get

$$T_{tr} = \frac{1}{\frac{1}{T_m} + \frac{1}{T_\beta} - \frac{1}{T_{SVN}}} \tag{16}$$

Expected battery life when transcoding is used with the buffered playback

Let B_{bf} be the average transfer rate of the buffered playback. Let T_{bf} be the expected battery life while movie v_1 is played back with the buffered playback. Let T_{on} be the time while the WNIC is turned on and the buffer is filled, and T_{off} be the time while the WNIC is turned off. We call $T_{on} + T_{off}$ the buffering period.

Let τ_{on} and τ_{off} be the time until the WNIC goes into the suspension mode from the normal mode and the time until the WNIC goes back to the normal mode from the suspension mode, respectively.

Let τ_{oh} be $\tau_{on} + \tau_{off}$. The value of τ_{oh} can be measured for each combination of a portable device and a WNIC. When there are i buffering periods during T_{bf}, $T_{bf} = i(T_{on} + T_{off} + \tau_{oh})$ holds.

We define n as follows.

$$n = (T_{on} + T_{off} + \tau_{oh}) \,/\, \left(\frac{T_{on}}{\beta(B_{bf})} + \frac{\tau_{oh}}{\beta(0)} \right)$$

Then, we get

$$\begin{aligned} E_0 &= T_{bf}(S_0 + V_1) + i\, T_{on} \frac{N_0}{\beta(B_{bf})} + i\, \tau_{oh} \frac{N_0}{\beta(0)} \\ &= T_{bf}(S_0 + \frac{V_0}{m} + \frac{N_0}{n}) \end{aligned} \tag{17}$$

During the buffered playback, variation of the data amount in the buffer (the maximum size is M bit) as time progresses is shown in Fig.2.

We get from Eq. 2,

$$T_{on} = \frac{M}{B_{bf} - b_1}\ , \quad T_{off} = \frac{M}{b_1} - \tau_{oh} \tag{18}$$

With

$$E_0 = T_n(S_0 + V_0 + \frac{N_0}{n}) \tag{19}$$

we get from Eqs. 3 and 4,

$$T_n = \frac{T_{SVN} T_{SV}\ n}{T_{SVN}(n-1) + T_{SV}} \tag{20}$$

Hence, we get, from Eqs. 3, 12, and 19,

$$T_{bf} = \frac{1}{\frac{1}{T_m} + \frac{1}{T_n} - \frac{1}{T_{SVN}}} \tag{21}$$

Thus, the battery life can be computed from values of parameters T_{SVN}, T_{SV}, T_{SN}, T_{base}, T_S, B_{bf}, τ_{oh} and values of r_0, f_0, b_0 of the movie v_0. Note that these values can be measured for each combination of a portable device and a WNIC.

3.2. Algorithm to Decide Battery-aware Parameter Values

The desired playback duration T must be less than or equal to T_S. Let R, F and B be the picture size, frame rate and bit rate of a movie sent from a content server.

Let r_1, f_1 and b_1 be the parameter values of the transcoded movie. Let M be the buffer size used in the buffering playback. Our method determines the values of r_1, f_1, b_1 and M for any $T < T_S$.

When a movie is transcoded to that with half picture size or frame rate, the bit rate can also be halved without serious quality degradation. Thus, we reduce the bit rate in proportion to the reduction ratio of the picture size and the frame rate.

Parameters of our algorithm are as follows.

1. Desired playback duration T
2. Parameters of the original video stream R, F, and B.
3. Terminal information consisting of (i) constant values (specific for each portable device) T_{SVN}, T_{SV}, T_{SN}, T_{base}, T_S, B_{bf} and τ_{oh}, and r_0, f_0 and b_0 of the movie v_0, and (ii) remaining battery amount $P(0 < P \leq 1)$, where $P = 1$ means that the battery is fully charged.
4. Preference information consisting of (i) allowable buffering delay D (in second) and (ii) priorities x, y and z for the video quality which corresponds to the picture size, the frame rate and the bit rate, respectively. x, y and z are given as integers between 0 and 2 where 0 is the highest priority and 2 is the lowest priority. Note that the quality with priority 0 is never degraded. At least one of x, y and z must be more than 0.

We show the algorithm to determine video parameters below.

1. Let r, f and b be $R/10$, $F/10$ and $B/10$ respectively [1].
2. $i := 1$
3. $r_1 := R$,$f_1 := F$,$b_1 := B$.
4. $M := Db_1$.
5. Determine T_{bf} using Eq. 21. Terminate and return values of r_1, f_1, b_1 and M if $T \leq T_{bf}P$.
6. $r_1 := R - irx$,$f_1 := F - ify$
   $b_1 = (B - ibz)(r_1 f_1)/(RF)$
7. if $r_1 < 0$ or $f_1 < 0$ or $b_1 < 0$, terminate since there is no solution.
8. $i := i + 1$, goto step 4.

For example, when $(R, F, B) = (49152, 24.0, 300)$, $(T_{SVN}, T_{SV}, T_{SN}, T_{base}, T_S) = (109, 182, 140, 148, 307)$, $(r_0, f_0, b_0) = (49152, 24.0, 300)$, $B_{bf} = 2.5$, $\tau_{oh} = 3$, $(x, y, z) = (1, 1, 0)$, $P = 1$ and $D = 80$, if $T = 200$, the values are reduced as follows until $T \leq T_{bf}$ is satisfied.

- $(r_1, f_1, b_1, M, T_{bf}) = (49152, 24.0, 300, 2.86, 157)$
- $(r_1, f_1, b_1, M, T_{bf}) = (44236, 21.6, 243, 2.32, 180)$
- $(r_1, f_1, b_1, M, T_{bf}) = (39321, 19.2, 192, 1.83, 204)$

4. Implementation and Experimental Results

We have implemented a video player for PDAs in C using Berkeley MPEG Player[7]. We have also implemented a transcoding proxy in C using MJPEG Tools [8].

4.1. Performance of Transcoding Proxy

In our transcoding proxy, we first derive a sequence of plain pictures in YUV format by decoding a MPEG1 stream, reduce the picture size and/or the frame rate of those pictures, and then reencode them into the new MPEG1 stream.

In order to evaluate the performance of our transcoding proxy, we have measured the processing time when a video (256x192 pixels, 24 fps, and 300kbps) is transcoded to the reduced quality video (176x132 pixles, 12 fps, and 75kbps) on a PC with general performance (Intel Pentium 4 2.40GHz, 1.0 GB RAM, Debian GNU/Linux 3.0, Kernel 2.4.18). As a result, for the original video whose playback duration is 261 seconds, our transcoding proxy could transcode it in 27 seconds. Thus, our implementation has sufficient performance for real-time transcoding.

[1] in general, $r, f\ b$ may be $r/m, f/m$ and b/m, respectively for any integer m.

Table 1. The actual values of the terminal information (SL-C700)

T_{SVN}	109 (min)
T_{SV}	182 (min)
T_{SN}	140 (min)
T_{base}	148 (min)
T_S	307 (min)
τ_{oh}	3 (s)
B_{bf}	2.5 (Mbps)

Table 2. The actual battery life and the predicted battery life

picture size (pixel)	frame rate (fps)	bit rate (kbps)	buffer size (MB)	actual battery life (min)	predicted battery life (min)
streaming playback of the video without transcoding					
256x192	24	300	–	109	109
buffered playback of the video without transcoding					
256x192	24	300	3.0	155	157
buffered playback of the video with transcoding					
256x192	24	150	1.5	185	201
256x192	12	150	1.5	210	224
176x132	24	150	1.5	225	225
176x132	12	75	0.75	274	257

4.2. Experimental Results

In order to examine the effectiveness of our power saving methods, we have carried out several experiments using PDAs (SHARP ZAURUS SL-C700) with IEEE802.11b wireless LAN card (WN-B11/CF,I-O DATA Device, inc.). Here, we have measured durations between the time when the battery was fully charged and the time when the battery was exhausted, while playing back video streams in various parameters.

The parameter values of the video v_0 is $r_0 = 256$x192 (pixel), $f_0 = 24$ (fps), and $b_0 = 300$ (kbps). The actual measurement of terminal information is shown in Table 1.

The actual battery life and predicted battery life using transcoding and buffered playback are shown in Table 2.

Table 2 shows that the transcoding and the buffered playback can extend battery life. We have confirmed that when the picture size and the frame rate is reduced to half, bit rate is reduced to one-third and the buffer size is set to 0.75 MB, our system can extend battery life up to 2.5 times. The difference between predicted battery life and actual battery life was within 9%.

5 Conclusion

In this paper, we proposed a low power video streaming method which uses a transcoding technique and a buffered playback mechanism to save power consumption. We defined a power consumption model on our proposed system to predict the battery life, and proposed a method to decide appropriate values for battery-aware parameters based on the model.

In this paper, we do not deal with the load fluctuation of a transcoding proxy and bandwidth fluctuation between a transcoding proxy and PDAs. However, if a transcoding proxy handles multiple users simultaneously, we need to consider those fluctuations. In the future work, we intend to explore possible approaches such as dynamic assignment of the transcoding proxy to an appropriate node in the network.

References

[1] Simunic, T., Benini, L., Acquaviva, A., Glynn, P. and Micheli, G. D.: Dynamic voltage scaling for portable systems, *Proc. of the 38th Design Automation Conf. (DAC2001)* (2001).

[2] Pering, T., Burd, T. and Brodersen, R.: Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System, *Proc. of Driven Microarchitecture Workshop, in conjunction with ISCA98*, pp.107–112 (1998).

[3] Lahiri, K., Raghunathan, A. and Dey, S.: Battery-Efficient Architecture for an 802.11 MAC Processor, *Proc. of the 2002 IEEE Int'l. Conf. on Communications (ICC2002)* (2002).

[4] Yip, K.W. and Ng, T.S.: Fast Power Control, Transmit Power Reduction and Multimedia Communications over WLANs, *Proc. of First Int'l. Conf. on Information Technology and Applications (ICITA2002)* (2002).

[5] Agrawal, P., Chen, J-C, Kishore, S., Ramanathan, P. and Sivalingam, K.: Battery power sensitive video processing in wireless networks, *Proc. IEEE PIMRC'98*, Vol. 1, pp.116–120 (1998).

[6] Lienhart, R., Holliman, M., Chen, Y.-K., Kozintsev, I. and Yeung, M.: Improving Media Services on P2P Networks, *IEEE Internet Computing*, Vol. 6, No. 1, pp. 73–77 (2002).

[7] The Berkeley MPEG Player. http://bmrc.berkeley.edu/research/mpeg/

[8] MJPEG Tools. http://mjpeg.sourceforge.net/