Control-Theoretic Dynamic Frequency and Voltage Scaling for Multimedia Workloads

Zhijian Lu, Jason Hein, Marty Humphrey†, Mircea Stan, John Lach, Kevin Skadron†
Dept. of Electrical and Computer Engineering, †Dept. of Computer Science
University of Virginia
Charlottesville, VA 22904, U.S.A.
{zl4j, hein, mircea, jlach}@virginia.edu {humphrey, skadron}@cs.virginia.edu

ABSTRACT

This paper describes a formal feedback-control algorithm for dynamic voltage/frequency scaling (DVS) in a portable multimedia system to save power while maintaining a desired playback rate. Our algorithm is similar in complexity to the previously-proposed change-point detection algorithm [19] but does a better job of maintaining stable throughput and is not dependent on the assumption of an exponential distribution of the frame decoding rate. For approximately the same energy savings as reported by [19], our controller is able to keep the average frame delay within 10% of the target more than 90% of the time, whereas the change-point detection algorithm kept the average frame delay with 10% of the target only 70% or less of the time executing the same workload.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: *design studies*

General Terms

Measurement, Design, Performance

Keywords

Feedback control, frequency/voltage scaling, low energy, power aware, high performance, control theoretic, multimedia, real-time

1. INTRODUCTION

Low power techniques in real-time (RT) embedded systems become more important as the performance expectations of these embedded devices continue to rise. RT embedded systems such as on-board satellite controls contain hard and soft RT deadlines and a strict limit on available battery power. A complicating factor for many applications is that tasks or service requests are often unpredictable, and current solutions for power aware computing cannot provide the guarantees necessary for RT. Over-provisioning is an appropriate solution for some systems but usually not for battery-operated systems where the battery has substantial cost or interferes with the desired form factor.

Closed-loop feedback control is an attractive way to attain soft-RT guarantees. Feedback control defines target metrics and error terms for the system being controlled, monitors the error, and continuously adapts the system to minimize the error. Feedback control can adapt to a wide range of behaviors and hence responds well to unanticipated workloads or behaviors. We use a simulated multimedia system to demonstrate the benefits of feedback-control. Our technique is motivated by the approach of Stankovic *et al.* [22], where a feedback control RT CPU-scheduling scheme is proposed; the design and performance evaluation is shown in [11]. In our feedback-based system, the CPU operating frequency and voltage are scaled based on the current frame throughput.

Contributions. Following [19], this paper models an MPEG portable multimedia-playback system with the content received over a wireless network. These multimedia systems require soft RT guarantees on frame delay to maintain the throughput necessary for avoiding choppy playback. We develop a formal feedback-control system to hold the CPU at the minimum possible operating frequency while still continuously adapting to provide the desired throughput. We evaluate the energy savings of our system using an analytic model for a variety of synthetic workloads representing exponential, uniform, and normal distributions of frame-arrival and decoding rates, and also real workloads on a Compaq iPAQ using StrongArm SA-1100 processor. This paper extends our previous work presented at the 2002 Workshop on Self-Healing, Adaptive and Self-Managed systems (SHAMAN) [14].

The rest of the paper is organized as follows. Section 2 describes related work in this field. Section 3 discusses our control-theoretic system, and Section 4 shows our results on both actual and synthetic workloads. Finally, we summarize our work in Section 5.

2. RELATED WORK

Circuit-level techniques have been a mainstay of power reduction for some years, but recently much research attention has focused on system-level techniques. The benefits of approaching power reduction at the system-level are typically synergistic with circuit-level techniques, and higher

CASES 2002, October 8–11, 2002, Grenoble, France.

design abstraction levels have more direct knowledge about the workload and can control large portions of the computer system accordingly. Yet for RT systems, performance should still be guaranteed even when power reduction techniques are in place.

Since task scheduling in computing systems is a key lever for both performance and power, it is a natural target for research on power-aware computing. There are two well studied power reduction techniques that have impacts on system scheduling [8]: DVS (dynamic voltage scaling) and DPM (dynamic power management). In DVS, different computational tasks are run at different voltages and clock frequencies while still providing an adequate level of performance. DPM aims to shut off system parts (inside or outside the CPU) that are not in use at any given time. Execution bandwidth throttling has also recently been proposed, in which the number of instructions fetched per cycle is reduced [17] or the frequency of fetch operations is varied [1, 21]. Structure resizing for power reduction has also been considered.

In the realm of power savings for multimedia workloads, *change point detection algorithm* based on DVS is introduced in [19]. This algorithm uses online statistical maximum-likelihood analysis to detect changes in aggregate behavior for a stream with an exponential distribution. They evaluate the algorithm using a system model that consists of an SA-1100 processor with an MP3 and MPEG workload. This work is similar to that of [16], which implements the workload of an H.263 video benchmark with frequency scaling on an SA-1100. The SA-1100 stalls execution when the clock frequency is adjusted; researchers noted a latency of 140 μs for each change in frequency setting. We use a similar simulation framework and workload to evaluate our control-theoretic scaling algorithm for both frequency and voltage scaling, the latter of which provides quadratic energy savings. Both frequency and voltage scaling are provided by most power-aware processors today. We explore both continuous and discrete DVS settings. When the number of discrete setting is large (e.g., 32 steps as described in [18]), we find no discernible difference between the performance of continuous and discrete DVS settings.

Recently, a new technique for saving energy in multimedia applications using architectural adaptation and frequency scaling was introduced in [7]. However, that technique uses profiling to predict energy per instruction and instructions per frame statistics. Architectural and frequency adaptations are then set to ensure frames meet their deadlines by adjusting the frequency to reduce slack time between frames. Our work features a real-time online technique using feedback control frequency scaling to provide energy savings.

The design of control systems is a mature field with a history dating back at least as far as the 1600s. Numerous textbooks exist that describe basic control principles, e.g. [4, 5]. Control-theoretic approaches have been applied to a variety of computer system design aspects outside the computer architecture realm, including CPU scheduling [12, 23], web server quality-of-service management [9, 13], inter net congestion control [6], and data migration [10]. At the circuit level, feedback control is used for voltage scaling [3] and current canceling for leakage control [25], but neither technique uses control theory to derive stability or to prove properties of the resulting system. The only use of *formal* feedback control theory that we are aware of in computer architecture literature is our own prior work on temperature regulation [21] and cache decay [24].

3. CONTROL SYSTEM DESIGN

In order to simplify our control system design, we model the multimedia decoding system as an M/M/1 queue, as suggested in [19]. Thus, the following equation holds:

$$T = \frac{1}{\mu - \lambda} \tag{1}$$

where T is the average delay for the frames, μ is the frame decoding rate and λ is the frame arrival rate.

Our goal is to adjust the CPU frequency and voltage such that the user-specified delay (referred to as the controller set-point) will be satisfied with the minimum amount of energy and the controller will stabilize quickly to provide a fast response time to variations in delay.

The basic architecture of our control system is illustrated in Figure 1. A generic control system architecture usually consists of an input set-point, a controller, a system or "physical plant" that is being controlled, and a feedback path. Our system includes the following components:

Input set-point: The desired average frame delay specified by the user.

Controller: An integral controller and its gain K_i. The output of the *integral* controller is the current (new) decision for the frequency scaling factor, which will go to the decoder system. At the beginning of the operation, we always let the frequency scaling factor equal unity, i.e, we run the decoder at the maximum speed. We force this initial value by adding a constant "1" to the output of the controller. We use a simple integral controller in our system because it can guarantee that the final output of the system will be equal to the set point when the system is stable. The value of K_i will be discussed later.

Controlled system: The multimedia frame decoder which accepts as its input the frequency scaling factor from the controller. A non-linear system model specified by Equation 1 is used to model the decoder and it will be linearized for controller analysis.

The actual frame arrival rate and decoding rate in the run time are unknown to the system and can change vary rapidly. We only assume that these values are less than 100 frames per second.

Feedback path: Because what we actually measure from the system is the average frame delay under the previous frequency scaling decisions, we model this feedback signal with a delay unit.

The controlled system is non-linear, but it can be linearized using first order Taylor expansion about the desired decoding rate μ by the following linear equation:

$$T(k+1) = T(k) - T_0^2 * (\mu * r(k+1) - \mu * r(k)) \tag{2}$$

where T_0 is the user specified delay and $r(k+1)$ is the frequency factor (a value between 0 and 1) adjusted by our controller at time $k+1$. For example, the decoder will run at its full speed when r is 1. This linear equation assumes that the controlled system works at operating points very

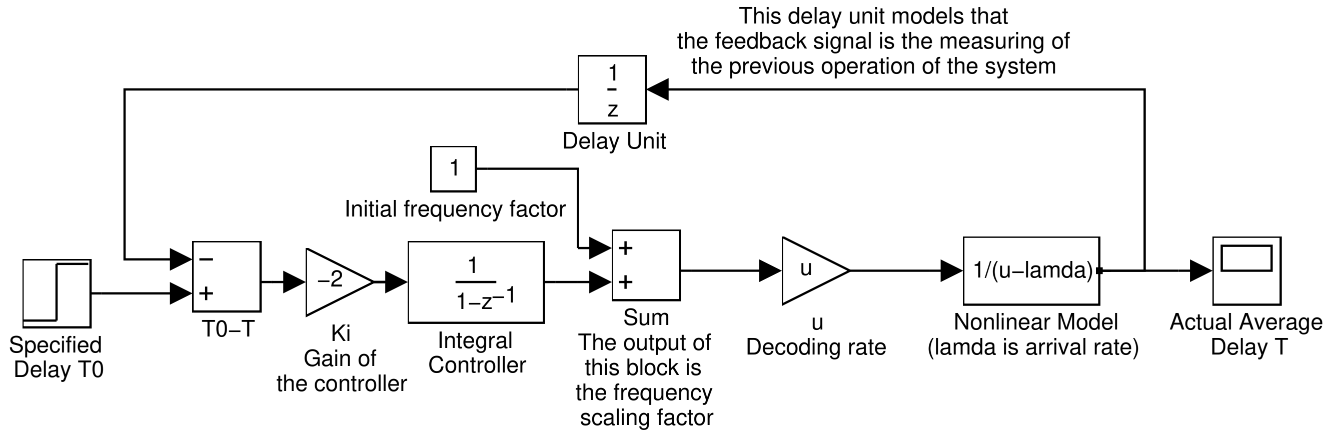

Figure 1: Basic architecture of our control system. All of the functional blocks are shown with the Z transforms of their respective transfer functions.

close to the set-point. As long as the stability of our control system is guaranteed, this assumption will hold. Note that this assumption is only needed for the control-theoretic analysis.

Using the system model shown in Figure 1 and substituting our linear model with the non-linear one, we can find the stability criteria for the whole system based on control theory: $-1 < 1 - |K_i|T_0^2\mu < 1$ or $|K_i| < \frac{2}{T_0^2\mu}$ where K_i is the gain of the integral controller. While a large value of K_i would make the system more responsive, the system would tend to be unstable. Given that in our system T_0 is about 0.1 second and the frame decoding rate is less than 100 frames/sec, and considering the sign of the gain value, we let $K_i = -2$. To show that the system avoids unstable oscillation, we use Matlab to simulate the system shown in Figure 1 with a step input signal. The step response of the system is shown in Figure 2.

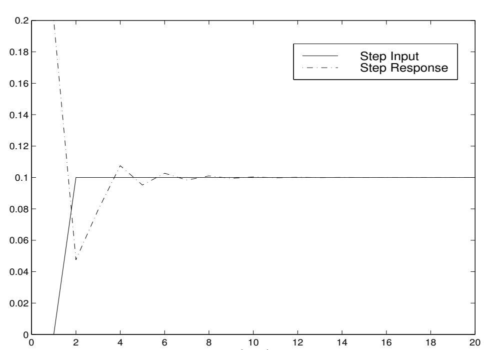

Figure 2: Step response of the system shown in Figure 1. Although we use the example frame arrival and decoding rates shown in Figure 1, other rates yield similar results.

As we can see from Figure 2, the system output converges to the set-point very quickly (after about 5 samples(iterations)) We then make use of this fast-converging property in our system to satisfy the RT constraints.

Besides the basic control architecture shown in Figure 1, we observe that the number of frames in the queue waiting to be decoded is a good indicator of future traffic. Thus, we feed the number of frames in the queue to a separate differential controller; the output of this controller will also be used to adjust the frequency scaling factor assigned to the decoder.

The feedback signal to our controller is the average delay over a certain number of previous frames (specified by our *window size*). Since Equation 1 holds true only for average values over a long time interval, larger window sizes can provide a higher-quality feedback signal (i.e., provide a better estimate of the actual average frame delay in the system). However, a larger window may increase the controller's reaction time to a changing situation (i.e., a different arrival or decoding rate). By simulation, we empirically determined that an average window size of 70 frames provides a good design trade-off. Our *sample* period is five frames (i.e., the controller will determine a new frequency factor every 5 frames), and frames in the same period will be decoded with the same frequency.

In order to make the best determination of the working frequency for the next sample period, we not only use the actual delay statistics from the previous frames in the window size, but we also use them as training samples in our control loop. At each sample, the controller will produce a new frequency factor based on the difference between the set-point and the actual average delay from the previous frames. We then back-calculate for each frame in the entire window how long that frame would have taken to decode had it been processed using the freshly-calculated frequency/voltage setting. The new average delay (a virtual value) is then fed to the controller again in an iterative relaxation process. Due to the fact that our controller converges quickly, it can arrive at a very good decision after only a few iterations in our implementation. There are two reasons for this approach. First, we believe that the frames in the next sample period will have very similar statistics to the previous frames in the window. Second, the controller can apply better decisions directly on the future frames, thus providing very rapid response to changes in the environment. This "virtual training" scheme makes the controller fast; otherwise, it would take many sampling periods for the moving average to reflect the new CPU frequency.

In order to keep the overall average frame delay as close to the user-specified set-point as possible, we use a counter to accumulate the difference between individual frame delay times and the set-point. This provides a global picture of how well the controller is performing and allows us to make the set-point adaptive. At each sampling point, if the controller finds that the previous average delay is much higher than the set-point, it lowers the actual set-point used for the next period to obtain better performance. On the other hand, if the previous average delay is in fact lower than the set-point, the controller increases the actual set-point to save more energy. This adaptive set-point scheme provides the best way to meet the user-defined requirements. This mechanism is equivalent to adding another control loop outside the whole system shown in Figure 1.

Though it seems that our scheme may consume a large amount of computation time and energy, the computation required for our controller is less than that of the change-point detection approach proposed in [19]. We compare the amount of computation needed for both algorithms in the average case. In the change-point detection scheme, the algorithm is executed for every new frame to be decoded. This algorithm assumes that there is a set of predefined arrival/decoding rates. For example, there may be 10 possible arrival/decoding rates. For each frame, the algorithm checks the previous 100 frames to identify the possible *change point* (i.e., the frame among the last 100 from which the arrival/decoding rates appears to have changed) by comparing the statistical maximum likelihood with a certain threshold. The amount of computation that this algorithm requires for each frame is about (including both arrival and decoding rate detection):

$2*10*10*\frac{1}{2}*$(2 adds + 3 multiplications + 1 logarithms) = 200 adds + 300 multiplications + 100 logarithms

In our control-theoretic scheme, the bulk of the computation is for the virtual training. Since our algorithm is invoked every 5 frames and (for our initial implementation) we use 5 iterations for virtual training, we have an average of one iteration per frame. The computation needed to calculate the average delay for the previous 70 frames when a new frequency is virtually applied is:

70*(2 multiplications + 5 adds) = 350 adds + 140 multiplications

Our later simulations show that the number of iterations can be even further reduced while maintaining performance.

Once the new frequency scaling factor is obtained, we scale the system voltage to the minimum that is required to maintain the new working frequency. The relation function between the working frequency and the respective minimum system voltage can be measured in advance for any system under study.

4. SIMULATION EXPERIMENTS AND RESULTS

We evaluate the performance of our control-theoretic approach on both actual and synthetic workloads.

4.1 Evaluation Using Actual Workloads

We implement our control-theoretic system on a Compaq iPAQ running an SA-1110 StrongARM processor. The iPAQ runs the Windows CE operating system. Running the control based model requires the scaling of core processor frequency, which is discussed in [2]. With the direct hardware access techniques discussed in [2], frequency scaling is accomplished by writing the desired frequency index to the Phase Locked Loop Configuration Register (PPCR). This allows for on the fly frequency scaling during program execution.

An open-source MPEG codec is provided by the MPEG Software Simulation Group (MSSS). Four MPEG-1 video files were used as input to the MPEG decoder to produce frame decoding rates for an actual workload. The four MPEG-1 videos (three 900 frame video files and one 1800 frame video file) yield frame rates of less than 1 frame/sec. This is much lower than typical MPEG frame rates of between 15 and 20 frames/sec. However, we expect lower frame rates from a device running Windows CE on a slow processor with a maximum speed of 221.2 MHz as well as a slower bus speed. Slow decoding rates are not expected from commercial MPEG players such as Windows Media Player, however commercialized MPEG players are highly optimized and source code is proprietary information not available to the public. Such commercial MPEG players provide a much higher and predictable frame rate. Possible future work is to obtain source code for a commercial multimedia player to obtain additional MPEG decoding rate trace data.

Using actual MPEG-1 trace files containing actual decoding rates, we tested our control-theoretic algorithm on the iPAQ device. Evaluating the efficiency of our control algorithm showed a total overhead of approximately 0.371 milliseconds per frame. Since this is less than 1 percent of a typical decoding time, which is normally around 40 milliseconds, we find that our control-theoretic approach is very efficient when running on an actual system.

Implementing our control-theoretic algorithm on such a system shows that our model is robust and capable of running on a handheld device with little overhead. Dynamically scaling frequency on a handheld device allows the portable system to have extended battery life while maintaining performance. A comprehensive evaluation of the real-world battery savings and quality of service characteristics on the iPAQ as a result of these approaches is the subject of future work; however, preliminary results indicate a strong correlation to the simulation results shown in the next section.

4.2 Evaluation Using Synthetic Workloads

For experiments on synthetic workloads, we used Matlab to generate traces matching the desired distributions (exponential, Gaussian, etc.) We have 1000 frames in each simulation run, and the statistics of the frames (i.e. arrival rate and decoding rate) may or may not change after every 250 frames.

Figure 3 shows the frequency adjustment for one simulation run. In this simulation, all frames have the same decoding rate–70 frames/second[1]. The arrival rate is reduced from 45 frames/second to 5 frames/second after the 250th frame. From the 750th frame, the arrival rate changes

[1]High decoding rates and arrival rates are used in this experiment to show the adaptive nature of the two approaches.

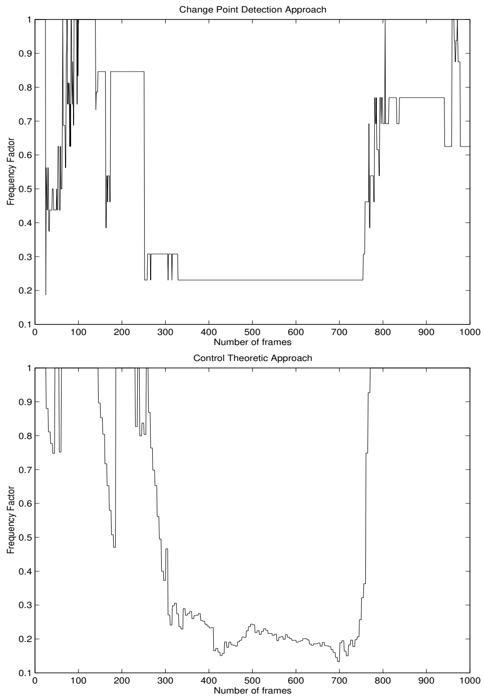

Figure 3: Frequency scaling curves for both algorithms in one simulation run.

sharply to 50 frames/second. The frequency scaling curves in Figure 3 show that both the change-point detection algorithm and our control-theoretic approach can adapt to the workload changes very quickly.

Mattavelli *et al.* [15] have shown that multimedia frame-decoding rates cannot be modeled by any simple distribution. In order to further evaluate the performance of the two approaches, we use different random distributions to create our synthetic workloads and divide them into 6 groups:

1. Both inter-arrival times and decoding times are exponentially distributed with high CPU utilization.

2. Both inter-arrival times and decoding times are exponentially distributed with low CPU utilization.

3. Both inter-arrival times and decoding times are exponentially distributed with hybrid (mixture of high and low) CPU utilization.

4. Inter-arrival times are exponentially distributed, and decoding times conform to Gaussian distribution, with hybrid CPU utilization.

5. Inter-arrival times are exponentially distributed, and decoding times are uniformly distributed, with hybrid CPU utilization.

6. Inter-arrival times are uniformly distributed, and decoding times conform to Gaussian distribution, with hybrid CPU utilization.

High CPU utilization means the CPU will mostly operate at a speed close to its fastest speed. On the other hand, low CPU utilization means the CPU need only operate at a slow speed in order to satisfy the delay requirement. For example, in Figure 3, the last 250 frames impose a high utilization on the CPU of the decoder, while the frames between 250th and 750th impose only a light workload for the CPU.

Each workload group is composed of 100 different simulation runs of 1000 frames. We measure, for each simulation run, the average delay, total energy, and frame-delay variance. We do not make measurements of the energy on the actual system due to the difficulties and inaccuracies of doing so on the iPAQ. Instead, our work adopts the approximate energy calculation proposed in [20], and it is used throughout our simulations. This approximate energy calculation provides the power consumption as follows:

$$E(r) = CV_0^2 T_s f_{ref} \left[\frac{V_t}{V_0} + \frac{r}{2} + \sqrt{r\frac{V_t}{V_0} + \left(\frac{r}{2}\right)^2} \right]^2 \tag{3}$$

where C is the average switched capacitance per cycle, T_s is the sample period of a DSP system, f_{ref} is the operating frequency at V_{ref}, r is the normalized processing rate (i.e., the frequency scaling factor in this paper[2]), V_t is the threshold voltage and $V_0 = (V_{ref} - V_t)^2$.

For a DVS system, C, T_s, V_t and V_0 are unchanged, and we can obtain the following reduced quadratic power consumption model [20]:

$$E(r) = r^2 \tag{4}$$

At the same time, for each workload group, we calculate how many simulation runs will lead to an average frame delay within $\pm 10\%$ of the user specified delay (delay set-point). Results for each simulation run from 3 workload groups are presented in Figure 4, and similar results are obtained from the other 3 workload groups. The statistics (average energy consumption and average delay standard deviation) for all 6 workload groups are summarized in Tables 1 and 2.

It is clearly shown in all six cases that our control-theoretic approach outperforms maximum likelihood detection in quality of service while providing comparable energy savings. With a minimal relative variance, our approach is able to provide smooth transitions to yield very consistent frame delays. Consequently, the average frame delay for the control-theoretic approach oscillates around the set-point of 0.1 seconds with minimal deviation.

Tables 1 and 2 show that our control-theoretic approach can guarantee the user specified set-point with very high probability. Energy consumption increases only slightly compared to the change-point detection algorithm. In contrast, the change-point algorithm tracks the user-specified frame delay rather poorly. Furthermore, notice that although our control system is designed with an M/M/1 queuing model as the system model, and that assumption is only required for control-theoretic analysis, so it still performs well in the cases where the decoding time is Gaussian or uniformly distributed. This demonstrates the robustness of the formal feedback-control approach.

Tables 3 and 4 illustrate examples of varying numbers of iteration cycles for the "virtual training" scheme. In the M/M/1 model and in the exponential/uniform model,

[2]We assume a linear relationship between the processing rate and the frequency scaling factor in our simulation, though it is not exactly linear in the real physical system

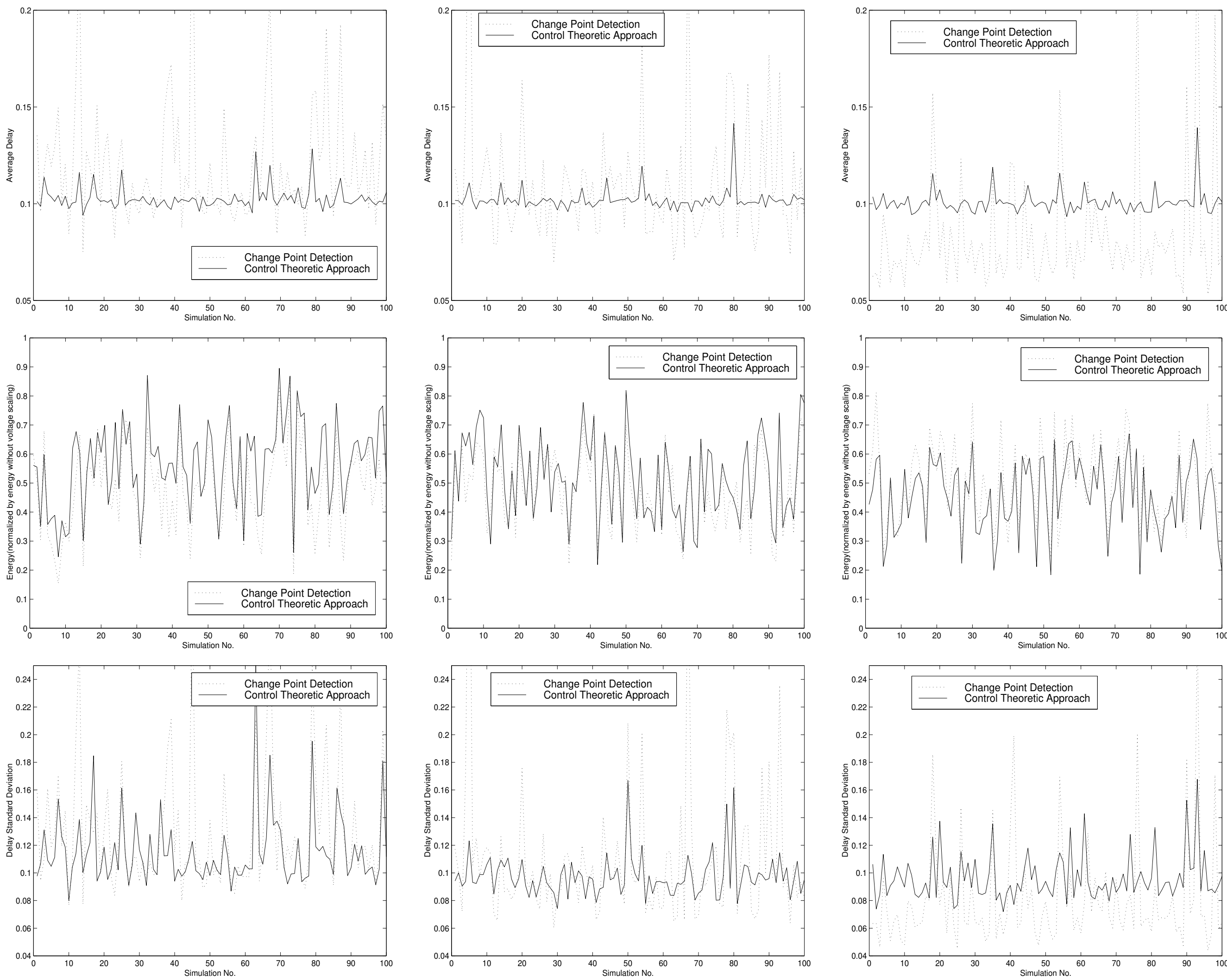

Figure 4: Performance comparison of the two approaches in terms of average delay, energy consumption, and delay variance for all simulation runs in three workload groups (left column: M/M/1 with hybrid CPU utilization; middle column: Exponentially distributed inter-arrival time and Gaussian distributed decoding time with hybrid CPU utilization; right column: Uniformly distributed inter-arrival time and Gaussian distributed decoding time with hybrid CPU utilization). The control-theoretic approach provides better timing guarantees with comparable energy savings. Similar results are obtained for other workloads including uniformly distributed decoding times.

Workload groups	Ratio of sample runs near set-point (+/-10%)	Average energy over all runs in the same group (ratio)	Average standard deviation over all runs in the same group (sec/frame)
Group 1	0.40	0.68	0.12
Group 2	0.71	0.23	0.11
Group 3	0.41	0.48	0.13
Group 4	0.38	0.47	0.11
Group 5	0.35	0.43	0.08
Group 6	0.04	0.50	0.08

Table 1: Performance summary for the change-point detection approach [15]. The energy column gives the energy consumed relative to no frequency/voltage scaling.

Workload groups	Ratio of sample runs near set-point (+/-10%)	Average energy over all runs in the same group (ratio)	Average standard deviation over all runs in the same group (sec/frame)
Group 1	0.90	0.73	0.11
Group 2	1.00	0.28	0.11
Group 3	0.92	0.56	0.12
Group 4	0.94	0.52	0.10
Group 5	0.97	0.42	0.08
Group 6	0.94	0.45	0.10

Table 2: Performance summary for our control-theoretic approach.

Iteration number	Average Frame Delay	Frame Delay Standard Deviation
1	0.099	0.100
2	0.102	0.099
3	0.102	0.093
5	0.102	0.092
8	0.102	0.095
10	0.103	0.097

Table 3: Impact of number of iterations of "virtual training" for a trace of M/M/1 Model with average delay set point = 0.1 sec.

Iteration number	Average Frame Delay	Frame Delay Standard Deviation
1	0.105	0.102
2	0.107	0.110
3	0.11	0.110
5	0.107	0.106
8	0.098	0.094
10	0.103	0.106

Table 4: Impact of number of iterations of "virtual training" for a trace of exponentially distributed inter-arrival times and uniformly distributed decoding times with average delay set point = 0.1 sec.

the average frame delay remains steady with little variation. The number of iterations ranges from 1 to 10. Therefore, the number of iterations in the "virtual training" back-calculation does not affect frame delay, and multiple iterations are used to achieve a responsive controller.

Frequency scaling is done in a continuous fashion, though we have briefly explored the method of quantized dynamic frequency scaling described in [18]. Frequency settings are linearly subdivided into 32 discrete levels in order to decrease the number of frequency transitions needed. We found that quantizing frequency scaling factors into discrete levels has a negligible effect on performance since frequency transition time is small compared to frame decoding time.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we describe a control-theoretic dynamic frequency/voltage scaling scheme for a simulated multimedia portable device. Our control design uses virtual training and an adaptive set-point to provide fast response and stable multimedia throughput, regardless of workload behavior/distribution. This system provides comparable energy savings to the change-point detection technique [19] but dramatically improves stability in average frame delay. It is also not dependent on the assumption of M/M/1 queuing behavior.

Overall, we have shown that formal feedback control is a robust and responsive way to control DVS settings for real-time response. The next step in this work is to fully integrate our controller into a QoS framework, using variance information to place the set-point so as to minimize jitter. More broadly, future work consists of the exploration of other applications of feedback control in power-aware computing.

6. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grant Nos. CCR-0105626, CCR-0133634, and a grant from Intel MRL. We would also like to thank T. Simunic, J. Pouwelse, and the anonymous reviewers for their helpful comments.

7. REFERENCES

[1] D.Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, pages 171–182, January 2001.

[2] I. Castillo. Introduction to Direct Hardware Access on Pocket PC. *http://www.columns.pocketnow.com*, September 2001.

[3] N. Dragone, A. Aggarwal, and L. R. Carley. An adaptive on-chip voltage regulation technique for low-power applications. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 20–24, July 2000.

[4] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems.* Addison-Wesley, third edition, 1994.

[5] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems.* Addison-Wesley, third edition, 1998.

[6] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. A control theoretic analysis of RED. In *Proceedings of IEEE INFOCOM*, April 2001.

[7] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *Proceedings of the 2001 International Symposium on Microarchitecture*, December 2001.

[8] N. K. Jha. Low power system scheduling and synthesis. In *IEEE Int. Conf. on Computer-Aided Design*, November 2001.

[9] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, June 2001.

[10] C. Lu, G. A. Alvarez, and J. Wilkes. Aqueduct: Online data migration with performance guarantees. In *Proceedings of the USENIX Conference on File and Storage Technologies*, pages 219–230, January 2002.

[11] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Design and evaluation of a feedback control edf scheduling algorithm. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, pages 56–67, December 1999.

[12] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Systems Journal*, 23:85–126, 2002.

[13] Y. Lu, A. Saxena, and T. F. Abdelzaher. Differentiated caching services: a control-theoretical approach. In *Proceedings of the International Conference on Distributed Computing Systems*, April 2001.

[14] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach and K. Skadron. Control-theoretic dynamic frequency and voltage scaling. In *Proceedings of the 2002 Workshop on Self-Healing, Adaptive and Self-Managed systems (SHAMAN'02)*, June 2002.

[15] M. Mattavelli and S. Brunetton. Implementing real-time video decoding on multimedia processors by complexity prediction techniques. *IEEE Transactions on Consumer Electronics*, 44(3):760–767, August 1998.

[16] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proceedings of the Conference on Mobile Computing and Networking*, July 2001.

[17] H. Sanchez et al. Thermal management system for high-performance powerPC microprocessors. In *Proceedings of COMPCON'97*, February 1997.

[18] G. Semeraro, et al. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings in the 2002 International Symposium on High-Performance Computer Architecture*, February 2002.

[19] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli. Dynamic voltage scaling for portable systems. In *Proceedings of the Design Automation Conference*, June 2001.

[20] A. Sinha and A. P. Chandrakasan. Energy efficient real-time scheduling. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, November 2001.

[21] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, pages 17–28, February 2002.

[22] J. A. Stankovic, C. Lu, S. H. Son, and G. Tao. The case for feedback control real-time scheduling. In *Proceedings of the IEEE Euromicro Conference on Real-Time*, June 1998.

[23] D. C. Steere, A. Goel, J. Gruenburg, D. McNamee, C. Pu, and J. Walpole. A feedback-driven proportion allocator for real-rate scheduling. In *Proceedings of the Third Symposium on Operating System Design and Implementation*, pages 145–158, February 1999.

[24] S. Velusamy, K. Sankaranarayanan, D. Parikh, T. Abdelzaher, and K. Skadron. Adaptive cache decay using formal feedback control. In *Proceedings of the 2002 Workshop on Memory Performance Issues*, May 2002.

[25] L. S. Y. Wong, S. Hossain, and A. Walker. Leakage current cancellation technique for low power switched-capacitor circuits. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 310–315, August 2001.