A Game Theoretic Approach for Power Aware Middleware*

Shivajit Mohapatra and Nalini Venkatasubramanian

School of Information and Computer Science
University of California, Irvine
mopy,nalini@ics.uci.edu

Abstract. In this paper, we propose a dynamic game theoretic approach for choosing power optimization strategies for various components(e.g. cpu, network interface etc.) of a low-power device operating in a distributed environment. Specifically, we model the energy consumption problem as a dynamic non-cooperative game theoretic problem, where the various components of the device are modelled as the players in the game that simultaneously consume a common resource(device battery power). An analysis for the Nash and social optima of the game is presented. We then introduce an adaptive distributed power-aware middleware framework, called "Dynamo", that incorporates the game theoretic approach for determining optimal power optimization strategies. We simulate the distributed game environment for proxy-based video streaming to a mobile handheld device. Our performance results indicate that significant energy savings are achievable for the device when the energy usage of the individual components achieve a social optima than when the energy usage achieves the strategic Nash equilibria. The overall utility of the system is measured both in terms of energy gains and the quality of video playback. Our results indicate that the device lifetime was increased by almost 50%-90% when compared to the case where no power optimization strategies were used, and 30-40% over device lifetime when Nash equilibrium is achieved; the overall utility of system for both types of equilibria were similar(utilities differ by $\leq .5\%$), indicating that the Nash equilibrium strategies tend to overuse the battery energy consumption.

Keywords: power optimization, game theory, power-aware middleware

1 Motivation

Limiting the energy consumption of low-power mobile devices has become an important research objective in recent years. The capabilities of these devices are limited by their modest sizes and the finite lifetimes of the batteries that power them. As a result, minimizing the energy usage of every component (e.g. CPU, network card, display, architecture etc.) in such devices remains an important

* This work was supported by funding from ONR MURI Grant N00014-02-1-0715 and NSF Career Grant ANI-9875988.

H.-A. Jacobsen (Ed.): Middleware 2004, LNCS 3231, pp. 417–438, 2004.

design goal and continues to pose significant challenges. These issues have been aggressively pursued by researchers and numerous interesting power optimization solutions have been proposed at various cross computational levels – system cache and external memory access optimizations [18], dynamic voltage scaling(DVS) [9, 7] of the CPU, dynamic power management of disks and network interfaces(NICs) [10, 4, 5], efficient compilers and application/middleware [20, 19] based adaptations for power management. Consequently, future generations of these low-power mobile devices will represent a new class of "*power-aware*" systems. These power-aware systems will be able to make the best use of the available battery power by adapting their behavior to the constraints imposed by their operating environments (users, network topology etc.). Additionally, components of these systems will be capable of multiple modes of operation for power management. Already, current wireless network cards have various power modes (sleep, transmit, idle etc.) and some CPUs (e.g. Transmeta's Crusoe) can be operated at various lower voltages(or frequencies). Moreover, the selection of the modes would be accomplished through various strategies that would control the aggressiveness of the power management for that component.

Interestingly, power optimization techniques developed for individual components of a device have remained seemingly incognizant of the strategies employed for other components. Therefore, increased research effort needs to be devoted to study the important issues involved in the interplay between the power management [25, 18] of the various components. While focussing their attention to a single component, researchers make a general assumption that no other power optimization schemes are operational for other components. Consequently, only the most aggressive forms of power management for individual components are investigated. We contend that unless a study is made of the trade-offs involved in the joint operation of the various components and the customizations/adaptations therein, the power gains (or performance) may turn out to be reductive instead of cumulative. For example, a cache optimization strategy for power optimization might adversely affect the performance of an aggressive DVS based algorithm, as the execution times of the tasks might be affected. Therefore, when multiple components are co-operating to effect power savings, the most aggressive strategies may not necessarily be the best ones. At a very high level, we view the system as a collection of components, that draw power from a common shared energy source (battery) and provide some utility in return. The overall utility of the system can be considered to be a function(e.g. sum, product etc. and is usually defined by the system designer) of the individual utilities of the components. We now need to solve the following problem: *how can we maximize the cumulative user experience (e.g. quality of video for multimedia applications) of the system while ensuring that the low-power device is operational for the longest time*? Fortunately, this problem is amenable to game theoretic analysis, which provides powerful tools for analyzing precisely such interactions.

Game Theory [1, 21] provides a set of tools to model interactions between agents with conflicting interests. For decades, game theoretic tools have been

used by economists and others to model economic agents such as firms and stock markets. Game theory typically assumes that all players seek to maximize their utilities in a perfectly rational manner. Economists have a hard time as human players are seldom perfectly rational. However, as in our case, when players are computational entities, it is reasonable to assume some notion of strong rationality (at least as far as computationally possible). Therefore, game theoretic analysis has also been widely used in the study of power control [17], flow control [3] and routing problems [23] in wireless networks. The purpose of our study is to use game theoretic analysis to tailor aggressiveness of power optimization techniques (for individual components), such that both the battery lifetime as well as the cumulative system utility are optimized.

2 Modelling Power Optimization as a Dynamic Game

To model a joint management strategy for power optimization, we must first identify the sources of power consumption. In modern mobile systems, there are three primary sources of power consumption: the CPU, the network interface and the display. At the architectural level, components such as caches, memory and logic gates are also driven by battery power. In this section, we present our view of the system and model the power management problem as a dynamic game. A basic introduction to game theory and some preliminary definitions are posted at *http://www.ics.uci.edu/˜dsm/dyn/prelim.pdf*. In a typical low-power system, we have multiple components jointly utilizing a resource (the battery) to which they all have access. In exchange for the extraction of some fraction of the resource, they provide some utility to the user of the system. As an example, for streaming media applications a measure of utility could be both the battery lifetime of the device and the application output quality as perceived by the user. The actual representation of the utility is in itself a rather hard research issue as it might contain both objective and subjective elements. We will revisit this topic in a later section. Moreover, in this case, the residual power of the device (battery) evolves through time according to the pattern of past component usage; note that the overall utility of a set of power management strategies is impacted as the residual power vanishes.

We can characterize the conjunctive operation of the various components of a low-power device as a non-cooperative dynamic game(Γ). We denote the primary power consuming components of the system as the set of players (P) in the game. Therefore $P = \{P_{cpu}, P_{net}, P_{display}, ...\}$ and let P_N denote the number of such players. Note that all the players concurrently draw energy from a common exhaustible resource(battery). We define the "*game environment*" at a period T as the current residual energy of the device(or battery) $= E_R^T \geq 0$, which evolves over time depending on the energy consumptions of the individual components of the system. The period identifies the frequency at which a sub-game is played and identifies the points in time at which the strategies can be re-evaluated. The strategy space S of each player is represented by an aggregation of all the power management strategies that are available for that player. For example, the strat-

egy space for the processor can be denoted as $S_{cpu} = \{S^0_{cpu}, S^1_{cpu}, S^2_{cpu}, ..., S^N_{cpu}\}$, where there are "N+1" independent power optimization strategies available for the cpu. These strategies could represent the various dynamic voltage scaling (DVS) algorithms suggested for slowing down the cpu under various conditions for energy gains. In general, this strategy space would include all power management strategies available for cpu slowdown. Additionally, we define a basic strategy denoted by S^0_{cpu}, which denotes a strategy that does not employ any power optimization technique for the cpu. Consequently, the power consumed by a basic strategy would be the maximum of all the strategies in the strategy space of that player. Similarly, the strategy spaces are defined for other players(components) in the game. We denote player P_i's ($P_i \in P$) energy consumption during a period T by C_{iT} (for strategy S^k_i); C_{0T} is assumed to be the energy consumed by the player under its "basic strategy", i.e. S^0_i, where $i = cpu, net, display, etc.$. It is natural to consider $C_{iT} \geq 0$ and that consumption gives player P_i a payoff or utility. The value of E_R (residual energy of battery) constraints the total amount of energy that can be consumed by the players, i.e. at every period T, it must be the case that

$$\sum_{i=1..P_N} C_{iT} \leq E^T_R \tag{1}$$

The amount of residual energy that would remain when each player plays its basic strategy (no power optimization) is given by $X_T = E^T_R - \sum C_{0T}$, However, when power optimization strategies are employed, each player generates an energy saving over the energy consumed by its basic strategy. We denote these energy savings as $\triangle^k_i$, where i is the player index {cpu, disk etc.} and k is the strategy used by the player. Therefore, the residual energy available in the period $T+1$ is ($E^{(T+1)}_R \geq X_T$) and is given by

$$E^{(T+1)}_R = E^T_R - \sum C_{0T} + \sum \triangle^k_i \tag{2}$$

Now if energy gain was the only measure of the utility, then maximizing $\triangle^k_i$ would maximize the utility. However in practice, the payoffs for power management strategies are influenced by a number of factors beyond the control of the players. The form factor of the device, the number and type of executing applications, energy gains, perceived user satisfaction, QoS guarantees, application response times etc. are all factors that could define the utility of a particular strategy. This makes defining an ideal utility function a very hard research problem. For our purpose, we define the utility for a particular strategy as a function of the energy savings from the strategy (can be measured) and the perceived user satisfaction (determined subjectively). From experience, we know that these two factors are somewhat in conflict. For example, if we slowdown the cpu for power savings, the response times of the applications will increase thereby reducing the perceived user satisfaction. In multimedia applications, a slower processing of video frames might cause a jitter. In general, a more aggressive power saving strategy tends to save more power, but might have a greater negative impact

on the user perception. Consequently, higher $\triangle_i^k$ may not directly translate to a higher utility. Our objective is to determine a value for $\triangle_i$ such that both the device lifetime and the overall utility of the device is maximized. A logarithmic function closely depicts such an utility function. In Sec. 4.2, we study the utility functions for the CPU and the network card and show that they can be approximated using logarithmic functions. We therefore define player i's utility function when it employs strategy S_i^j as $log(C_{iT})$. Moreover, the amount of power saved by any strategy can be expressed as a function of the residual energy of the device at the time the strategy is employed; therefore we can define $\triangle_i^k = f(E_R^T)$ for each player. We can say that the power optimization strategies regenerate(in some sense) some of the energy that would otherwise be used up with the basic strategies. We therefore assume that the residual energy for the next period can be expressed as $E_R^{(T+1)} = f(X_T)$. This characterization lends the game to be analyzed as a classical game theory problem called the tragedy of the commons [2, 15]. Note that duration of this potentially infinite game can be restricted by setting a threshold battery energy level until which the game is played. This level can be selected such that minimum energy requirements for the device components are reserved.

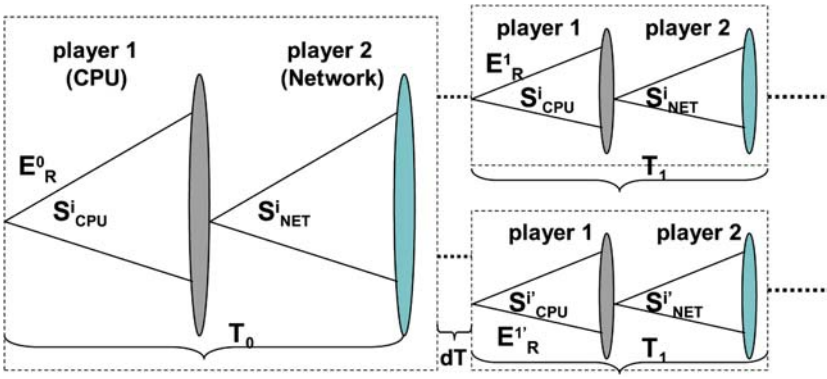

Fig. 1. Extensive form for a 2 player game(CPU,Network)

The extensive form of the game (with 2 players) is shown in Fig. 1. Initially, the residual energy of the low-power device is assumed to be E_R^0. At this point, both the CPU and the network interface card(NIC) choose their power management strategies indicated by S_{CPU}^i and S_{NET}^i. The game is played in accordance with these strategies for the period T_0. At the end of the period the strategies are again re-evaluated depending on the residual device power. We assume that the evaluation takes a small time δT as shown in the figure. At this point, both players decide on the new strategies and the game continues over period T_1. Note however, that depending on the initial strategies chosen the game could take one of many courses (2 of them are depicted in the figure). The oval represents the range of the strategies that can be employed by the players. This game continues indefinitely until either the battery drains out of power (inevitable) or the device is stopped.

2.1 Game Analysis

By designing the conjunctive component power management for low-power devices as a non-cooperative dynamic game (as above) makes it amenable to several different types of game theoretic analysis. In this section, we investigate the game from the perspective of Markovian strategies that provide Nash equilibrium for each period. This is representative of the current research approaches, where each player(component) unilaterally chooses its strategies with the objective of maximizing its own utility. We also present a social optimality analysis wherein the objective is to maximize the conjunctive utility of the system. This is representative of the approach we suggest. A brief comparison of the analytical results is made and its implications to our overall problem is discussed. Using our analysis, we can answer the following questions – how does the residual energy of the device y_t evolve over time? Does strategic interaction of the components lead to persistent overuse of the battery resource?

Before we present the analysis, we redefine some of the notations presented in the last section for easier readability. Let the "game environment" (residual energy of device) at the beginning of period t be $y_t \geq 0$. We denote player i's energy consumption due to the adopted strategy in period t as $c_{it} \geq 0$. Furthermore, in the event that the players attempt to consume energy in excess of the current residual energy, we assume that the total amount is split equally among the players. We perform our analysis on a two player game (CPU, network interface card) for power/utility management for two components). In low-power devices, the CPU and the network card account for a significant percentage of the overall energy consumption. In the case of the LCD displayo, the main energy drain comes from the backlight, which has a predefined user setting and therefore has a limited degree of controllability by the system through various strategies. This remains a subject of ongoing research and therefore we do not include the backlight in our analysis. In the two player game: i) the analysis is much simpler and is easier to understand and (ii) it is representative of the general "N" player game, that is a "N" player game can be analyzed in much the same manner. Fortunately, for our purposes, the maximum number of players is three (N=3) (considering the LCD display and the fact that most handhelds do not have disks).

Furthermore, as the analysis is quite complex for a two player game [2, 15], we show the analysis for specific forms of the utility function and the energy regeneration functions. An analogous analysis is possible for other types of functions. Without loss of generality, we assume that player i's utility from consuming c_i amount of energy in any period is given by $Blog(c_i)+C$, and that $y_{t+1} = A.{X_t}^D$, where A,B,C and D are constants. We simply denote the players by subscripts 1 and 2. More specifically, we use the functions $log(c_i)$ (B=1,C=0) for the utility function and $y_{t+1} = A.\sqrt{X_t}$ $(D = \frac{1}{2})$ to improve readability of the analysis.

2.2 Social Optimum Analysis

From an analogous economic game theory standpoint, social optimality is defined as: In a society of two individuals having simultaneous unrestricted access

to a common resource, how should each player extract the common resource such that this society of two individuals remain as "happy" as they can be. In our case, the "happiness" is represented by the joint utility of the players with the common resource being the battery energy. Therefore to derive the social optimality solution we need to consider the sum of the two players' utilities – and maximize it. This can be achieved through analysis by backward induction. Suppose to begin with, there are exactly 2 periods (as shown in Fig. 1). If we are in the last period with residual energy y, then we need to solve

$$Max\ \{log(c_1) + log(c_2)\} \tag{3}$$

where $c_1 + c_2 \leq y$. In order to maximize utility, all the available residual energy should be used up at this period; that is it must be that $c_1 + c_2 = y$. Hence, the maximization problem can be written as

$$Max_{c_1}\ \{log(c_1) + log(y - c_1)\} \tag{4}$$

Using the first order condition for maximization (1^{st} order derivative = 0) we get $\frac{1}{c_1} = \frac{1}{y-c_1}$; that is consumption by both the components should be equal ($c_1 = c_2$ and equal to $\frac{y}{2}$). Consequently, each components socially optimal utility when there is one period left and the residual energy is y, is given by $V^1(y) = log\frac{y}{2} = log(y) - log(2)$. This can be written as $log(y) + B(1)$, where B(1) stands for the constant -log(2). Let us now consider the penultimate period (i.e fold back the tree in Fig. 1). Clearly, when there are 2 periods left the socially optimum energy consumption is found from solving the following problem:

$$Max\ \{log(c_1) + log(c_2) + 2\delta V^1[A(y - c_1 - c_2)^{0.5}]\} \tag{5}$$

where $c_1 + c_2 \leq y$ and δ is the discount factor. In our case the discount factor is important because as the device runs out of power, the energy resource gets more valuable, and therefore so do the utilities. Since, $V^1[A(y - c_1 - c_2)^{0.5}] = log[A(y - c_1 - c_2)^{0.5}] = log(A) + \frac{1}{2}log(y - c_1 - c_2) + B(2)$; we can write the problem as

$$Max\ \{log(c_1) + log(c_2) + \delta log(y - c_1 - c_2)\} \tag{6}$$

where $c_1 + c_2 \leq y$, where we have suppressed the additive constants log(A) and B(2), as they do not affect the optimal choice. Again, the first-order conditions for maxima are obtained by equating the derivative of the above equation to zero. We have $\frac{1}{c_1} = \delta[y - c_1 - c_2]^{-1}$ and $\frac{1}{c_2} = \delta[y - c_1 - c_2]^{-1}$. Since, the expressions are identical, it must be that the two consumptions are equal. Using the above equations it follows that the common consumption is $\frac{y}{2+\delta}$. Note that the energy consumption is less than it is when there is only one period left. After collecting the terms, the socially optimal utility for a component can be written as $(1 + \frac{\delta}{2})log(y) + B(3)$, where B(3) is a compilation of constants.

Now we consider the case when there are more than 2 periods. Instead of solving the general case right away, let us do one more step of induction to see if there is a solution pattern. Now, suppose there are three periods of resource usage. In the first period we have the following problem to solve:

$$Max\ \{log(c_1) + log(c_2) + 2\delta V^2[A(y - c_1 - c_2)^{0.5}]\} \tag{7}$$

where $c_1 + c_2 \leq y$. Now, by substituting for V^2 and by suppressing all the (irrelevant) constants we can rewrite the last expression as

$$Max\ \{log(c_1) + log(c_2) + \delta(1 + \frac{\delta}{2})log(y - c_1 - c_2)\} \tag{8}$$

where $c_1 + c_2 \leq y$. Proceeding similarly as before we can get the first-order conditions for this problem as $\frac{1}{c_1} = \delta(1 + \frac{\delta}{2})[y - c_1 - c_2]^{-1}$ and an identical expression for c_2. It can be shown that the socially optimal consumption equals $\frac{y}{2}(1 + \frac{\delta}{2} + \frac{\delta^2}{4})^{-1}$ and the socially optimal utility for each player is of the form $(1 + \frac{\delta}{2} + \frac{\delta^2}{4})log(y) + A(3)$, where A(3) is a compilation of the constants. At this stage a pattern is clearly observed. Similarly, the analysis for "T" remaining periods is present in Table 1.

Table 1. Energy consumption for various remaining periods

periods remaining	consumption (fraction of y)
1	$\frac{1}{2}$
2	$\frac{1}{2(1+\frac{\delta}{2})}$
3	$\frac{1}{2(1+\frac{\delta}{2}+\frac{\delta^2}{4})}$
T (conjecture)	$\frac{1}{2[1+\frac{\delta}{2}+...+(\frac{\delta}{2})^{T-1}]}$

Using the above conjecture, we now know the equilibrium consumptions for the game for T periods. Note that in an infinite period model, we can get this identical consumption function (call it $c(y)$) by taking the limit of the optimal consumption as T$\rightarrow \infty$. Since, $1 + \frac{\delta}{2} + ... + \frac{\delta}{2}^{T-1} + ... = \frac{1}{1-\frac{\delta}{2}}$, we can say that

$$c(y) = \frac{1 - \frac{\delta}{2}}{2} y \tag{9}$$

Based on this optimal energy consumption rule an optimal power management strategy can be executed for each component. In a later section, we will discuss some component based power management strategies and how we profile such strategies for various components.

2.3 Best-Response(Nash) Equilibrium Analysis

We now present a parallel analysis in a strategic (rather than social) setting. Here the assumption is that the players are consuming the battery resource unilaterally. Therefore, each player (component) will only consider its own utility and seek the strategy that maximizes this utility. Much like the social optimality analysis, the game equilibrium can be solved by backward induction. We present a similar analysis for a two player game. As before, suppose we are in the last

period with residual energy y. At this point, all the energy can be consumed. Hence, the stage sub-game equilibrium is one where each player's actual consumption is $\frac{y}{2}$. Consequently, each components equilibrium utility is given by $W^1(y) = log(\frac{y}{2}) = log(y) + B(1)$, where B(1) is a constant (= -log(2)). Let us now fold the tree back to consider the penultimate period. When there are two periods left, player 1 faces the following best-response problem:

$$Max\ log(c_1) + \delta W^1[A(y - c_1 - \theta y)^{0.5}] \tag{10}$$

where $c_1 \leq (1-\theta)y$, δ is the discount factor and θ is the fraction of the resource that player 2 is expected to consume in the first period . Note that we have assumed that the consumption of player 1, $c_1 \leq (1-\theta)y$. Otherwise, we know that there will be no consumption for either player in the last period. Since, $W^1\{A(y - c_1 - \theta y)^{0.5}\} = log(A) + \frac{1}{2}log[(1-\theta)y - c_1] + B(2)$, we can rewrite the problem as

$$Max\ log(c_1) + \frac{\delta}{2}log[(1-\theta)y - c_1] \tag{11}$$

where $c_1 \leq (1-\theta)y$ and the constants are suppressed. Applying the first order condition we have $\frac{1}{c_1} = \frac{\frac{\delta}{2}}{(1-\theta)y - c_1}$. Therefore, the best response consumption is given by $(1 + \frac{\delta}{2})c_1 = (1-\theta)y$. If we write the consumption as a fraction of the residual energy at that time - that is, if we write it as $b(\theta)y$ - then it follows that

$$b(\theta) = \frac{1-\theta}{1+\frac{\delta}{2}} \tag{12}$$

If we do a similar analysis for player 2, we get the symmetric equilibrium condition, wherein each player has the same consumption, and the rate is such that it is the best response to itself. Therefore, $b(\theta) = \theta$. Put differently, the extraction rate $\frac{1}{2+\frac{\delta}{2}}$ is a symmetric equilibrium. As before, after collecting the terms, the equilibrium utility when there are two remaining periods, W^2, can be written as $(1 + \frac{\delta}{2})log(y) + B(3)$, where B(3) is a constant. After substituting the formula for W^2, it can be shown that the first period best response problem for player 1 is

$$Max\ log(c_1) + \frac{\delta}{2}(1 + \frac{\delta}{2})log[(1-\theta)y - c_1] \tag{13}$$

where the previous conditions for the variables hold. Solving for the first order condition we get $\frac{1}{c_1} = \frac{\frac{\delta}{2}(1+\frac{\delta}{2})}{(1-\theta)y - c_1}$. We get the symmetric consumption level for each player equal to $\frac{1}{2+\frac{\delta}{2}+\frac{\delta^2}{4}}$. As with social optimality we can generalize the solution for T periods as follows: when there are "T" remaining periods, the energy consumption fraction of each player is given by $\frac{1}{2+\frac{\delta}{2}+...+(\frac{\delta}{2})^{T-1}}$. In the infinite period model, the equilibrium consumption function (call it $c^*(y)$), will be given by the limit of the equilibrium consumption as $T \to \infty$. Since $2 + \frac{\delta}{2} + ... + (\frac{\delta}{2})^T + ... = 1 + \frac{1}{1-\frac{\delta}{2}}$, we can say that

$$c^*(y) = \frac{1-\frac{\delta}{2}}{2-\frac{\delta}{2}}y \tag{14}$$

Using the above equilibrium consumption, a best response strategy can be chosen for each component.

Discussion: From the above analysis we observe that when each component (player) employs its power optimization strategies unilaterally, there is a possibility of overuse of the battery resource. As mentioned earlier, such one-sided decisions do not necessarily translate to the highest overall utility for the system. Comparing the two consumption functions: the socially optimal function $c(y)$ (eqn 9) and the strategically optimal function c^*_y(eqn 14), we see that

$$c(y) = \frac{1 - \frac{\delta}{2}}{2} y \ < \ \frac{1 - \frac{\delta}{2}}{2 - \frac{\delta}{2}} y = c^*(y) \tag{15}$$

The equation holds as $(2 - \frac{\delta}{2}) < 2$. It can be concluded that the strategic equilibrium are suboptimal. While theoretically it has been proved that the social optimal is better than the strategic optimal, it is a challenge to design a system that can facilitate such optimal battery usage. In the next section, we present a middleware framework that can be effectively used for optimal use of the system battery resource.

3 The Dynamo Middleware Framework

In the previous section, we presented a theoretical analysis for optimized power consumption for a generic set of components and their power management strategies. However, in practice, the options available for power optimization are limited by type of low-power devices used and the context of the applications. For example, a cpu slowdown strategy that slows down the cpu by 70%(say) may not be feasible for multimedia applications(as frames cannot be decoded in time); again a handheld without a network card need not be optimized for that component. Therefore, we need to conduct a case-specific analysis for a given environment and device context. Furthermore, in our case, it is important to reevaluate the strategies used for various components as the game environment evolves (network/device conditions dynamically change). A distributed adaptive middleware framework designed for cross-level power optimization is a natural choice for performing such an optimality analysis. The system architecture for such an adaptive middleware framework (called Dynamo) is depicted in Fig. 2. A prototype implementation of the framework is presented in section 5.

In Fig. 2, the lowest level shows the various hardware components targeted for power optimization. The driver interfaces and the power optimization strategies for the various components are available at the operating system layer. A battery monitor provides higher layers with realtime information on the current residual battery level of the low-power device. Dynamo consists of a lightweight middleware runtime layer that executes on the device and provides an API interface for dynamically deploying power management strategies for the various components. Additionally, the framework contains a more heavy weight component that can execute on a network node (e.g. a proxy server) and performs

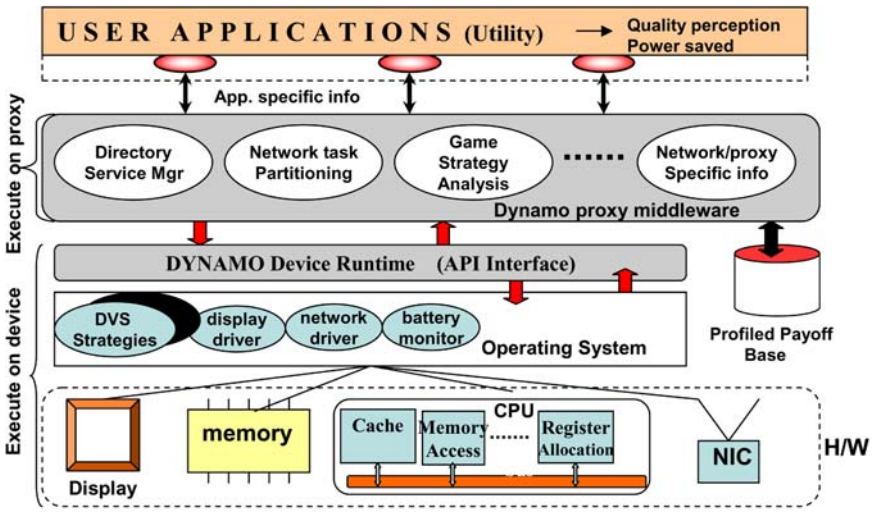

Fig. 2. The Dynamo middleware architecture

the game theoretic analysis remotely using a distributed protocol; As the more computationally expensive game theoretic analysis is shifted onto a distributed proxy, the middleware on the low-power devices can have a lightweight footprint. By using this model, the middleware can exploit knowledge of the local device state (e.g. residual power levels stored & updated at a directory service) and global state (e.g. network congestion, node mobility etc.) that can be available at the proxy to dynamically select optimal power optimization strategies for the components. We assume that the middleware has at its disposal a knowledge base of the strategy space and corresponding utility functions for each of the components. Such a knowledge base can be created by extensively profiling (or using research literature) each component and its various strategies under different operating conditions. Additionally, the middleware can implement various policies that affect the analysis of the strategic interaction of the components. For example, the middleware can fix the number of periods for which the game is played (or if the game is infinite) using various policies(e.g. constant,infinite etc.); dynamically modify the game environment incase there is a sudden drift in the battery energy level, assign value of the discount factor(δ) and set the threshold energy level before the start of the game. By using a distributed approach, much of the computationally expensive analysis is moved away from the low-power device to a network entity (proxy). Furthermore, the proxy is better suited to make dynamic global adaptations because it has information of the global state that would be unavailable at the device. The communication overheads in this approach are minimal as the proxy communicates with the device only when power management strategies for individual devices need to change.

The high level algorithm employed by the middleware for determining the socially optimal energy strategies are presented in Fig. 3. Fig. 3(a) presents the

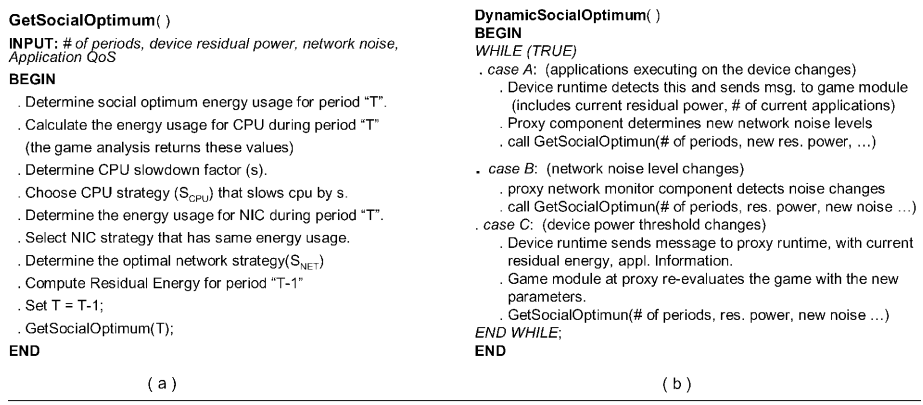

Fig. 3. High-Level Algorithm used for determining the Social Optimum Strategies

algorithm for a static scenario where parameters of the power management game are fixed. Given a residual power of the device, the number of applications and application QoS requirements and a constant network noise level, the algorithm can be used to determine strategies that achieve a social energy consumption equilibria. In a dynamic scenario, some or all of the game parameters can change randomly. The middleware can implement a dynamic algorithm(Fig. 3(b)) that detects these changes in application load, network noise levels and diminishing device power levels and repeatedly executes the static equilibrium algorithm continuous adaptation.

4 Performance Evaluation

We adopt a two pronged approach to evaluate the performance of our framework. First, we use profiled results to simulate the game environment and compare the performance of the game strategies. In Sec. 5, we present a prototype implementation of the middleware framework on a Linux based system. In this section, we focus on the results of our simulations in the context of video applications.

In our simulations, we consider two system components(CPU & NIC) for power optimization. We measure the energy consumption and overall utility of the system and the individual components, when the components consume energy according to "social" and "best-response" equilibria conditions. A comparison is made with the energy consumption of the baseline condition in which no optimizations are made for either the CPU or the NIC. We use a streaming video player as the user application executing on the device. Streaming video applications are ideal for our simulation, as they heavily use both the cpu and the network. The values used in our simulations are based on our extensive work [18] in profiling the power consumption characteristics of streaming video onto handheld computers. The next section presents the details of the simulation environment and a discussion on the strategy spaces used for both components.

Table 2. CPU and Video burst length configurations for ideal energy and performance gains

Video Quality	Cache (Size,Assoc)	Voltage	Original Energy	Optimized Energy	Savings	Video Bursts (in secs)	Power Saved (Watts)
Q1(Highest)	8,8	1	1.29	0.76	47.5%	2.3	0.925
Q2	8,8	1	1.09	0.64	47.8%	3.5	1.0
Q3	8,8	1	0.95	0.56	48.0%	4.6	1.04
Q4	32,2	0.9	0.54	0.26	57.6%	4.85	1.05
Q5	32,2	0.9	0.48	0.23	57.8%	6.8	1.08
Q6	32,2	0.9	0.42	0.20	58.0%	14.5	1.12
Q7	8,8	0.9	0.29	0.14	57.3%	17.5	1.13
Q8(Lowest)	8,8	0.9	0.24	0.11	57.5%	17.0	1.12

4.1 Simulation Environment

We model our low-power device after a Compaq iPAQ 3650, with a 206Mhz Intel StrongArm processor, with 16MB ROM, 32MB SDRAM. The iPAQ is equipped with a Cisco 350 Series Aironet 11Mbps wireless PCMCIA network interface card for communication. The streaming video application is modelled after the Pocket Video Player available for Windows CE. Table. 2 present sample values for optimized network and cpu operating points for videos of different qualities. We then identify the strategy spaces for the CPU and the NIC for the above device.

CPU: Instead of identifying individual CPU strategies for power optimization, we assumed that the speed of the processor can be varied continuously from the minimum(S_{min}) to the maximum(S_{max}) supported CPU speeds for the device, and normalize the values such that the operating range varies from $[\nu_{min}, 1]$, where $\nu_{min} = \frac{S_{min}}{S_{max}}$. We then use the commonly used energy model presented in [14] to calculate the power P as a function of "*slowdown factor*(ν)".
$P = f(\nu) = 0.284 \cdot \nu^3 + 0.225 \cdot \nu^2 + 0.0256 \cdot \nu + \sqrt{311.16 \cdot \nu^2 + 282.24 \cdot \nu} \times (0.0064 \cdot \nu + 0.014112 \cdot \nu^2)$
We assume that the strategy space for the CPU is the set of strategies that can can vary the CPU speed between the minimum and maximum supported speeds. However in practice, these values will be discrete slowdown factors supported by the CPU, so an approximation to the closest theoretical slowdown factor needs to be chosen. This can be achieved through an operating system API interface available to the middleware. Also using the above model, we can determine a CPU slowdown factor that corresponds to a particular energy consumption level.

NIC: In [4, 18, 22], it is demonstrated that if video packets are buffered and sent to the device in bursts, then the NIC card on the device can be transitioned from the "*active*" to the "*sleep*" mode, thereby saving significant power. As the energy saved for a network card is proportional to the amount of time it spends in the "*sleep*" mode, the energy consumption of the NIC is dependent on the burst sizes used for transmitting packets [18]. Note here that "burst size" refers to the number of seconds of video payload that can be buffered and sent to the device

in one burst over the network. While large bursts sizes can cause significant savings in power, they cause higher packet drop rates and buffer overflows at the wireless access point resulting in a significant drop in perceived video quality. We assume the strategy space for the network card as the set of strategies that set the burst sizes of video transmissions (in secs) to continuous values in the range between 1 second to 150 seconds. Note that for each burst size chosen there is a unique value for the energy consumed by the NIC. In the next section, we use our empirical studies from [18] to derive the utility functions for the CPU and the network card.

4.2 Utility Functions

In this section, we describe how we identify the utility functions for the various strategies used for the power management of the CPU and the NIC. Recall that we define the utility function for a strategy as a function of both the power consumed (essentially equal to power saved) and the satisfaction as perceived by the user. For video applications, we assume that the user perception is directly related to the quality of the video(described by frame rate, frame resolution and bit rate of the stream) as in [18]. Fig. 4 shows how the normalized power savings and the perceived video quality varies with the cpu slowdown factor. Clearly the power savings increase as the cpu is slowed. However, the user perception remains at the highest level till the cpu is slowed by about 48%. Subsequent reduction in cpu speeds causes a drop in the video quality, due to frame deadline misses. Fig. 5 shows the actual utility curve for the video as the sum of the curves in Fig. 4, plotted against the cpu energy usage. A curve fitting technique is then used to determine an approximation of this curve. We determine that the utility for the video application can be specified as a function of the power consumption as follows: $Utility = 0.0408Log(Power) + 1.369$, with a R^2 value of 0.0197. Fig. 6 demonstrates how video quality(based on % of pkts dropped) and the power savings of the network card vary with the packet burst sizes in seconds. Clearly, the power savings improve when the burst sizes increase. However, as the burst size becomes larger than 1.48 seconds, packets start getting dropped at the wireless access point. As a result, there is a perceived drop in the user perceived video quality. Fig. 7 shows the actual utility curve for the video against the burst size. Fig. 8 plots the power savings of the NIC versus the video burst size used. Using a curve fitting method can specify the network card power savings as a function of the packet burst size as follows: $y = 0.1909Log(x)+0.4139$. Using the above strategy spaces and utility functions for the CPU and the NIC, we now present the results for the overall energy savings achieved when the CPU and the NIC operate under conditions of "social" and "best-response" equilibria.

4.3 Experimental Results

We used our simulator to determine the energy consumption of the device under both static and dynamic application loads and network noise levels. In both

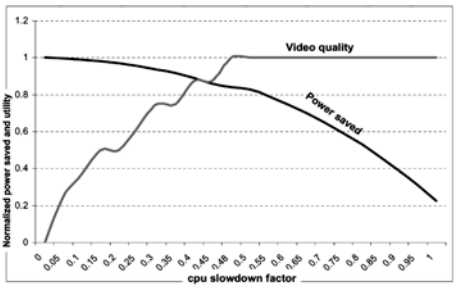

Fig. 4. power & video quality (cpu)

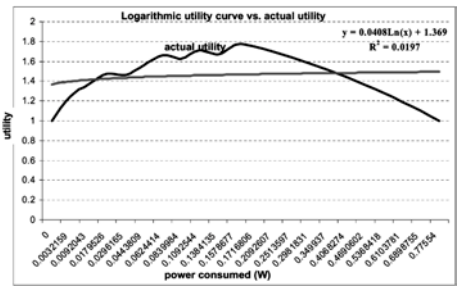

Fig. 5. cpu power vs. utility

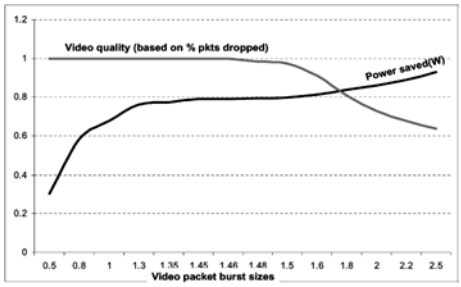

Fig. 6. power & video quality (NIC)

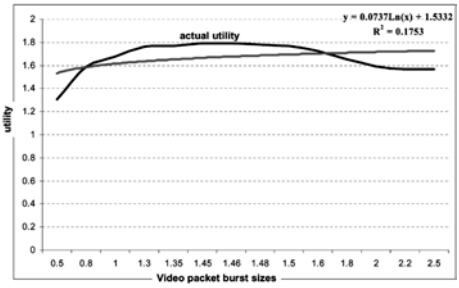

Fig. 7. video burst size vs. utility

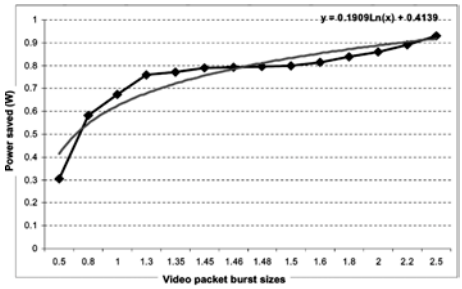

Fig. 8. burst size vs. power saved

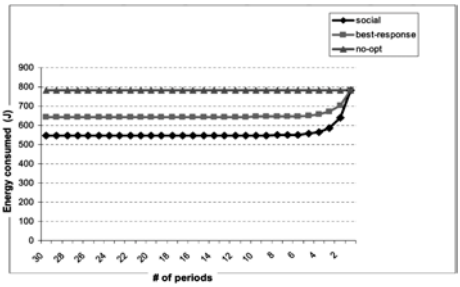

Fig. 9. total energy consumption

cases, we estimated the energy consumption characteristics of the device under three different operating (social,nash and no-optimization) conditions of the CPU and the NIC. For the static measurements, we assumed that a multimedia application (video player) playing streaming video on the device with a static network noise level. The energy consumptions were calculated assuming a fixed number of repetitions of the game while a single application executed on the device. In the dynamic case, we randomly started and stopped a set of applications as well as varied the network noise levels randomly. As a baseline condition, we estimated the energy consumption when no power optimization strategies were used. Under this assumption,the CPU operates at its maximum speed and the

network card is in the "active" state at all times. Next we measured the energy consumption of the device when the components used strategies that achieved the "social" and the "best-effort" equilibria respectively.

We first present the results of our simulation of the static case. Fig. 9 shows the total energy consumed by the device under the above three conditions, assuming the initial lifetime of the device to be 90 minutes and considering 30 repetitions(T=30) of the two-player game and a discount factor(δ) of 0.95. It is seen that the the overall energy consumption for the social optima is the lowest. However, both the best-response and the socially optimal energy consumptions are significantly less than the energy used when no power optimizations are in place. As seen in the analysis earlier, the social equilibrium tends to consume all the available energy in the last period of the game, therefore at that point its energy consumption equals that of the no optimization case. This is because we considered a finitely repeated game(T=30). However, at T=30, clearly the residual energy available at the device would be the maximum when the components consume energy in accordance with the social optimality condition.

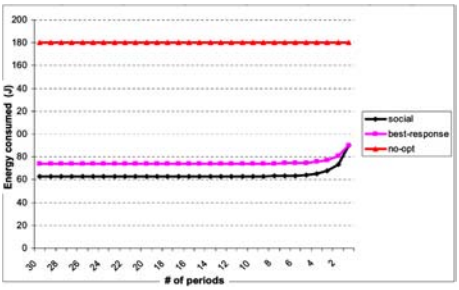

Fig. 10. CPU energy consumption

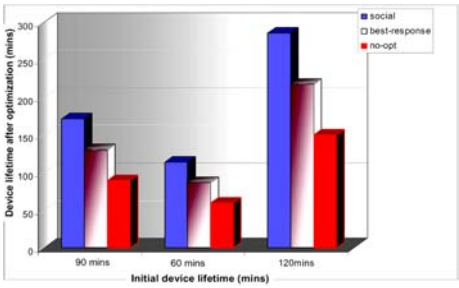

Fig. 11. Gain in device Lifetime

Fig. 11 shows the lifetime of the device for various initial values for residual energy. As seen from the figure, the device lifetime is significantly increased using the socially optimal strategies for both the CPU and the network. This is expected as less battery energy is drawn under this condition. Fig. 10 compares the energy consumed by only the CPU under the two equilibria. It is seen that the CPU consumes 13% less energy when it operates at a social equilibrium than when it operates under the best-response equilibrium. Fig. 12 compares the normalized video quality levels achieved by the strategies used for the CPU. Clearly, there is very little difference(≤ 0.1) in the normalized quality of video attained by the two equilibrium conditions for the cpu. However, much lower energy consumption levels are attained for the social equilibrium strategies. Note that the utility for the no-optimization case is much less than either of the above techniques as it consumes significantly more energy with possibly a slight increase in the user perceived quality.

In the dynamic case, we used a set of 6 applications and randomly started and stopped the applications and randomly varied the network noise levels. Fig. 13

shows the energy consumption characteristics of the CPU as the dynamic adaptation is performed for both social and Nash equilibrium conditions. The small frequent spikes in the graph indicate the points at which the application load (no. of applications) on the device changes. Fig. 15 shows the corresponding energy consumption plot for the NIC. For both the CPU and the NIC, the social equilibrium strategy tends to consume lesser energy than the Nash equilibrium strategy. The overall energy consumed over time for both the equilibrium conditions and the baseline case is shown in Fig. 14. The total energy consumption is much lesser for the socially optimal strategy.

Finally, in order to compare the dynamic adaptation with the static case, we started out by executing eight applications on the device and computed the static social and Nash equilibrium energy consumptions. Then we randomly stopped applications one by one and performed dynamic adaptation for the new application load. Intuitively, as the number of applications are reduced we should be able to reduce the equilibrium energy consumption dynamically, while still maintaining the same perceived quality. Fig. 16 shows the residual energy of the device over time for static and dynamic social equilibria. Clearly, significant amount of residual energy is saved over time when dynamic adaptation is performed. In conclusion, as the number of applications decrease randomly, dynamic adaptation can increase the overall lifetime of the device. On the other hand, as the number of tasks increase, dynamic adaptation can provide a better quality.

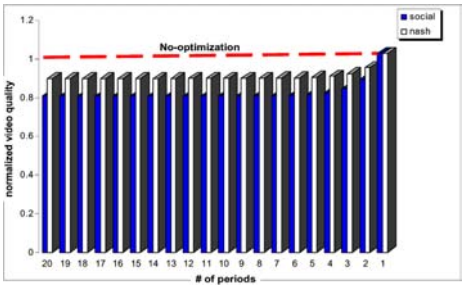

Fig. 12. Quality Comparison (cpu)

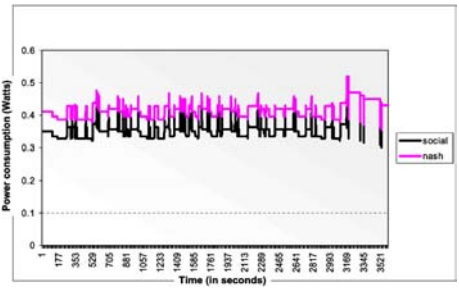

Fig. 13. CPU power vs. time

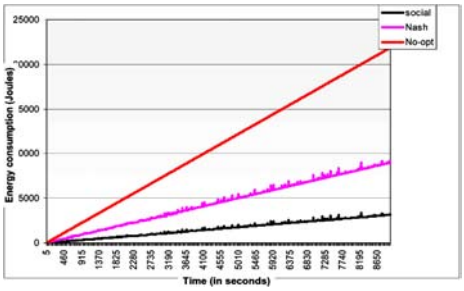

Fig. 14. Total Energy over time

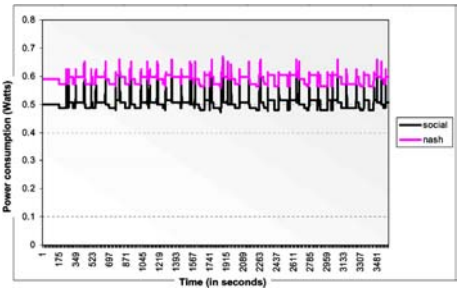

Fig. 15. Network power vs. time

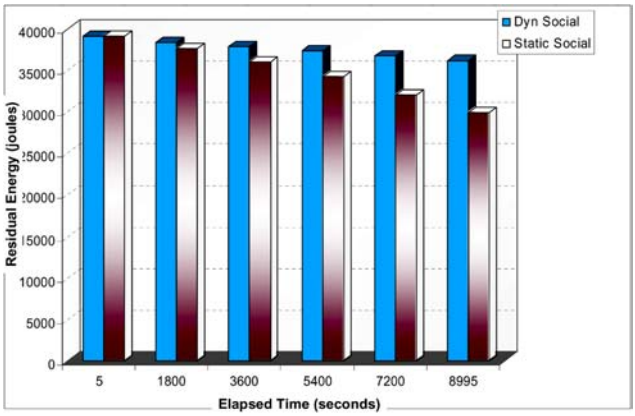

Fig. 16. Dynamic vs. Static Adaptation

4.4 Summary of Results

We compared the performance of socially optimal power consumption strategies with the strategies that achieve strategic equilibrium and the ones that implement no energy optimization under both static and dynamic load conditions. Under static conditions, the energy consumed for the socially optimal strategies was lesser(about 20J for every repeated game) than the energy used by the Nash strategies. The device lifetime was considerably increased(around 80-90% over no optimization strategies and 20-40% over Nash strategies) with only a slight decrease in the quality of the video application. We showed that a dynamic algorithm that adapts to changing application load and network noise levels using strategies that provide social optima provide significant energy gains over Nash strategies. Finally, we show that in situations where the application load changes over time, a dynamic algorithm performs better than a static algorithm. It was observed that the number of repetitions of the game(T) and the discount factor(δ) had little impact on energy usage levels of the components.

5 Prototype Implementation

We have implemented a prototype of the Dynamo middleware framework. The hardware platform for our implementation is the Sharp Zaurus (model SL5600) running the Linux operating system. It uses an Intel 400MHz Xscale processor and has 32MB of SDRAM and 64MB of protected flash memory. The Xscale processor can operate at various frequencies ranging from 100MHz(0.85V) to 400MHz(1.3V).As our proxy, we use a Windows XP desktop system with a 2.4 GHz processor and 512MB of RAM and a 40 GB disk. The handheld used a Cisco 350 Series Aironet 11Mbps wireless PCMCIA network interface card for communication. We use a National Instruments PCI DAQ board to sample voltage drop across the iPAQ at 200K samples/sec. The streaming video application is modelled after the freely available VLC media player for Linux.

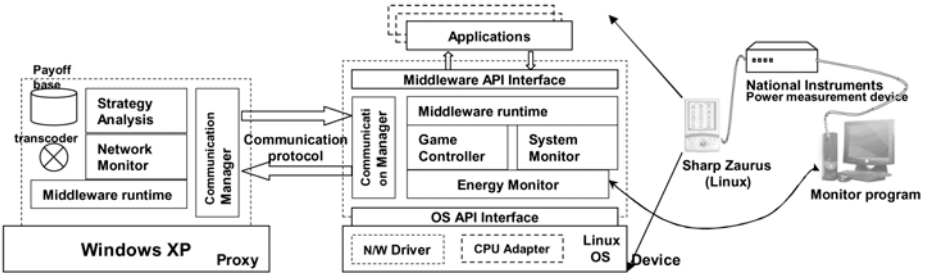

Fig. 17. Prototype Implementation in Dynamo

Fig. 17 shows the integration of the game analysis into the Dynamo middleware architecture. The middleware on the device includes four primary components – the *game controller*, the *system and energy monitors* and the *communication manager*. The middleware provides an API interface for applications to specify the QoS requirements and to change QoS requirements for dynamic adaptation. The *game controller* is used to specify the details of the game analysis and to set/modify the dynamic game parameters. The *system monitor* monitors the resource usage of the system and notifies the runtime of changes (e.g. change in the number of applications). The *energy monitor* communicates with the PCI DAQ board as well as interfaces with the low-level OS APIs to monitor the energy usage of the CPU and the network interface. The power-aware API (PAAPI) library for Linux is used to adjust the operating frequency of the CPU. The *communication manager* defines the middleware communication protocol and communicates with the proxy middleware. It uses UDP over IP for communication and a well defined structure interface for exchange of control information. On the proxy the Dynamo middleware uses a module to perform the strategy analysis using an profiled payoff base. A network monitor module maintains an updated state information on the overall congestion level. This information is used by the middleware to adapt the network traffic to the device. Fig. 18 lists a limited set of middleware API used at the low power device and briefly describes each function. It also presents the energy overheads of using the middleware framework and the communication overheads.

Middleware API	API Description
• *GetApplicationQoS( )* • *SetApplicationQoS( )*	-These functions are used to assign QoS (e.g. video quality) and application specific parameters (WCET, utilization etc.)
• *SendDeviceInfo( )* • *ReceiveControlInfo( )*	-Send device specific information to the proxy -Receive control information from the proxy
• *GetSystemInfo( )* • *GetBatteryInfo( )* • *SetCPUFreq( )* • *SetNetworkIdleTime( )*	-Get the current system state information (#apps, utilization) -residual Battery energy -Set the CPU frequency level -Set the idle time for the network
• *SetGameParameters( )* • *SetThresholdEnergy( )*	-Set the game specific parameters -Set a threshold energy level to account for other components

Energy Overhead (Avg.)	
Middleware Framework	0.6 Watts
Communication (over 10 min. period, 30Bytes • Send & Receive	0.34 – 0.4 W
Video Playback	6.05 Watts

Fig. 18. Prototype Middleware API and Initial results

Discussion: Note that the middleware needs to utilize a set of operating system API to achieve some of the low-level functionality. While a number of low-level power management knobs exist for handheld devices, currently many have to be statically configured (e.g. backlight intensity levels). Again, current Xscale processors provide only frequency scaling and it would be desirable to have voltage scaling as well. Similarly, the network interface cards have multiple low duty-cycle operating modes. However, support for dynamically exploiting these at higher levels (middleware and application) are still very limited. We are noticing a growing trend in the exposing of system level knobs to applications and middleware through well defined API interfaces, and have designed our framework to incorporate and exploit future enhancements to APIs at the OS and architectural levels. We also concur that some of our approaches can be better incorporated at the OS level, as it currently has a higher degree of control over hardware power management. The discussion of the low-level API is outside the scope of this work.

6 Related Work

The mathematical theory of games was first introduced by Neumann and Morgenstern [13] in 1944. Since then, game theory [21] has evolved into an important tool for analysis of conflict situations and has found invaluable application in the analysis of numerous social and economic conflict models. Traditionally, game theory has been used in computer science for development of computer games (e.g. chess) and in the areas of artificial intelligence. More recently, several interesting research efforts have applied game theoretic analysis to wireless communication systems [17], flow control [3] and routing [23]. In cellular systems users desire to have a high signal to interference (SIR) ratio at the base station (for low error rate, and reliability) coupled with the lowest possible transmit power (for longer battery life). A high transmit power used by a user can increase interference for other users thereby lowering their SIR. This might lead to other users to increase their transmit powers. Game Theory has been extensively used for analysis of such communication systems [17, 16, 8]. The use of game theoretic analysis for dynamic power management of disks has been suggested in [12]. In our work, we have used game theory to analyze the power management of various components in a low-power system, where we propose that the individual power management strategies for the various components should be chosen such that a socially optimal equilibrium condition is achieved. This is in accordance with the classical game theory problem called the "tragedy of the commons" [2, 15, 21].

On the other hand, power management for the individual components for low-power devices have been aggressively researched. Dynamic Voltage Scaling [9, 7] for saving energy consumption of CPUs have been extensively studied. At the application and middleware levels, the primary focus has been to optimize network interface power consumption [10, 4, 5]. A thorough analysis of power consumption of wireless network interfaces has been presented in [10]. Chandra et al. [4]

have explored the wireless network energy consumption of streaming video formats like Windows Media, Real media and Apple Quick Time. In [22], Shenoy suggests performing power friendly proxy based video transformations to reduce video quality in real-time for energy savings. They also suggest an intelligent network streaming strategy for saving power on the network interface. Caching streams of multiple qualities for efficient performance has been suggested in [11]. The GRACE project [25] claims the use of cross-layer adaptations for maximizing system utility. They suggest both coarse grained and fine grained tuning of parameters for optimal gains. In [24], the authors enhance the OS to support process groups which consist of a set of closely related/dependent processes. Coordination between the architecture(cache) optimizations, network and application adaptations through an adaptive middleware framework has been used in [18] to optimize power and utility for multimedia applications. In [19], a middleware framework that partitions reconfigurable middleware components between a low-power device and proxy for improving the costs of computation and communication is presented. Energy efficient battery management strategies have been extensively studied by Rao et al. [6].

7 Conclusions and Future Work

In this paper, we presented a dynamic game theoretic approach for choosing power optimization strategies for multiple components that draw energy from a common resource, the battery. We modelled the components as players in a non-cooperative game and determined how each component should draw battery power. We evaluated two techniques – one in which the components employ strategies that aim to maximize the overall utility of the system (social optimum) and another in which each component uses a best-response strategy for maximizing its own utility. Our performance results indicate that strategies that achieved a socially optimal energy usage provided the maximum energy savings and with similar utility values. We therefore conclude, that in a multi-component system, strategies for each component should be chosen such that they attain a socially optimal energy usage pattern. As an extension of our current efforts, we plan to employ game theoretic analysis for optimizing power consumption and performance of low-power devices by exploiting the knowledge of the distributed environment. It would be interesting to study a more distributed adaptation scheme involving multiple proxies and devices. We also plan on investigating the impact of optimizations on non realtime applications such as browsers and text editors etc..

References

1. A.J.Jones. *"Game Theory: Mathematical Models of Conflict"*. Ellis Horwood, 1980.
2. R. Amir and N. Nannerup. Information Structure and the Tragedy of the Commons. June 2000.
3. C.Douligeris and R.Mazumdar. "A game theoretic approach to flow control in an integrated environment with two classes of users". In *Computer Network Symposium*, April 1988.

4. S. Chandra. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. In *MMCN*, January 2002.
5. S. Chandra and A. Vahdat. Application-specific Network Management for Energy-aware Streaming of Popular Multimedia Formats. In *Usenix Annual Technical Conference*, June 2002.
6. C. F. Chiasserini and R. Rao. Energy Efficient Battery Management. In *IEEE Infocom*, March 2000.
7. K. Choi, K. Dantu, W.-C. Chen, and M. Pedram. Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder. In *ICCAD 2000*, 2002.
8. D.J.Goodman and N.B.Mandayam. "Power control for wireless data ". In *IEEE Personal Communications*, April 2000.
9. E.Chan, K. Govil, and H. Wasserman. "Comparing algorithms for dynamic speed-setting of a low-power cpu". In *Proc. of MOBICOM*, November 1995.
10. L. Feeney and M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an ad hoc Networking Environment. In *IEEE Infocom*, April 2001.
11. J. Flinn and M. Satyanarayanan. Energy-Aware Adaptations for Mobile Applications. In *SOSP*.
12. S. Irani, S. Shukla, and R. Gupta. "Competitive analysis of dynamic power management strategies for systems with multiple power saving states". In *DATE*, 2002.
13. J.V.Neumann and O.Morgenstern. "Theory of Games and Economic Behavior". In *Princeton University Press*, 1944.
14. P. Kumar and M. Srivastava. Predictive Strategies for Low-Power RTOS Scheduling. In *ICCD*, 2000.
15. D. Levhari and L.Mirman. The Great Fish War: An example using a Dynamic Cournot-Nash Solution. In *Bell Journal of Economics*, 1980.
16. A. B. MacKenzie and S.Wicker. "Game Theoretic approaches to distributed power control in cdma wireless data networks". In *IEEE Globecom*, November 2001.
17. A. B. MacKenzie and S.Wicker. "Game Theory in Communications: Motivation, explanation and application to power control". In *IEEE Globecom*, November 2001.
18. S. Mohapatra, R. Cornea, and et.al. "Integrated power management for video streaming to mobile handheld devices". In *ACM Multimedia*, 2003.
19. S. Mohapatra and N. Venkatasubramanian. PARM: Power-Aware Reconfigurable Middleware. In *ICDCS-23*, 2003.
20. B. D. Noble, M. Satyanarayanan, D.Narayanan, J.E.Tilton, and J. Flinn. Agile Application-Aware Adaptation for Mobility. In *SOSP*, October 1997.
21. P.K.Dutta. *"Strategies and Games, Theory and Practice"*. MIT Press, Cambridge, MA, 2001.
22. P. Shenoy and P. Radkov. Proxy-Assisted Power-Friendly Streaming to Mobile Devices. In *MMCN*, 2003.
23. T.Roughgarden and E.Tardos. "How bad is selfish routing?". In *IEEE Symposium on Foundations of Computer Science*, 2000.
24. W. Yuan and K. Nahrstedt. Process Group Management in Cross-Layer Adaptation. In *MMCN*, January 2004.
25. W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems. In *MMCN*, January 2003.