

Resource Access Pattern Mining for Dynamic Energy Management

Dinesh Rajan, Christian Poellabauer, Nitesh Chawla
{dpandiar, cpoellab, nchawla}@cse.nd.edu

Department of Computer Science and Engineering,
University of Notre Dame, USA.

Abstract. *Energy management mechanisms in resource constrained environments such as mobile systems are typically based on the current status of various system parameters such as utilization and quality of service requirements. Any knowledge gained about future system resource usage and other significant dynamics in the system can further lead to increased optimizations in such mechanisms. Future knowledge about system resource utilization and dynamics can be derived from two separate sources: a) the information maintained in the system about its current utilization and resource accesses and, b) the application specific behavioral patterns of currently active applications. This work targets at employing known efficient data mining techniques to exploit the latter class of information in achieving a suitable application behavior model that can provide accurate predictions of future application specific resource accesses and usage. Such formulated application behavioral patterns containing information about their resource usage, termed Resource Access Patterns, are then employed in a novel dynamic energy management scheme for a mobile system network resource. This work also shows that a data mining approach is beneficial in energy management decisions as it can capture all the underlying intricate interactions and dependencies that exist among the active applications and between the resources available in the system. The key components of this work can thus be summarized as: a) employing a data mining approach to characterize application behavior and, b) designing a novel energy management policy for a network interface device based on information about its future usage derived from the formulated application behavioral model.*

1 Introduction

In recent years, mobile computing has highly permeated into the daily lives of its users. At the same time, their use in highly hostile terrains such as rain forests, battlefields, disaster affected areas for performing critical tasks such as monitoring (enemy positions), searching (for survivors) and other strategic efforts (unmanned attack, rescue operations) have greatly increased the need to integrate autonomic and self-adaptive mechanisms into such systems. However their increased use and extension to such common and intricate tasks have been constrained by the limited energy resources available at their disposal. Several energy management policies and techniques [2, 12] have been proposed and implemented in addressing this constraint.

Typical energy management schemes are often based on the current value of system parameters such as processor utilization and quality of service requirements such as deadlines. Existing energy management mechanisms can further be enhanced by incorporating any knowledge of future system utilization, resource usage and other significant dynamics such as task scheduling into their schemes. Future knowledge about such system parameters can be obtained either from the information maintained in the system about its execution environment as a whole (eg., /proc interface in Unix based systems), or from the currently active applications themselves (eg., by parsing through their source codes).

Future knowledge about application specific behavior provide significant advantages over any general system level information about the future. This is because they offer better insights and higher accuracy on predictions about likely system resource accesses and their usage during execution of the

applications. As a result of the above benefits, this work aims to exploit the application specific information by employing data mining techniques to further enhance resource level energy management policies. Application run-time behavior can be derived using two distinct methods: a) *online* run-time monitoring of application execution, operations performed and their system resource accesses and usage and, b) compile-time analysis of application code providing *offline* static information about behavior of different execution regions in the application. An online approach offers more up-to-date information on application behavior and other dynamics (time spent, data size used during resource accesses) involved during application execution. Our work involving an online monitoring mechanism adds a new dimension to existing efforts in this direction by employing data mining concepts to model the behavior (in terms of resource accesses and usage) of common applications. The modeled behavioral patterns of applications are then used in predicting future resource access patterns which form the underlying basis of a novel energy management scheme for a network interface device in energy constrained mobile systems.

Data mining techniques besides providing a robust and automated methodology also offer the following benefits:

- i. A non-intrusive online approach for access to application behavior characteristics is achievable through the use of a data mining mechanism.
- ii. Data mining approaches inherently integrate the highly intricate dependencies that exist among all applications and between the various resources active in a computing system into the formulated models. This would help in achieving efficient prediction schemes thereby providing highly optimized energy management policies.
- iii. Additionally, a data mining approach allows formulation of models combining application-specific knowledge with the more universal system level knowledge thereby combining the advantages of two different methods.

It is well known that the secondary resources of network devices and disk drives together consume as much as a third or more of the total energy consumption of mobile computing systems [4, 9]. For this work, we limit ourselves to the resource of network interfaces due to the following factors: a) The energy consumption of modern network interfaces for mobile systems has increased drastically in keeping with the higher performance benefits offered. On the other hand, disk drives have been optimized for both energy consumption and higher performance and, b) Modern mobile systems are increasingly being used for communication oriented tasks such as media streaming, video conferencing etc.

In our work, the information about applications' resource accesses and run-time actions is gathered (as traces) during their execution using a custom built online system monitoring tool. Data mining concepts are then employed in mining and extracting the appropriate data of interest concerning resource accesses and usage. A causal relationship between the accesses made by an application to various resources is then established and a model of the pattern capturing this relationship among the resource accesses (termed 'Resource Access Patterns' or RAP) is built. The access patterns arranged sequentially in time model the applications' behavior with regard to resource accesses. Based on the past history of the pattern of accesses made by an application, a fair prediction of its future course of action and resource accesses is achieved. These predicted access patterns provide the basis for the proposed novel energy management scheme that adapts between the various energy optimization states (sleep, active) available for a network interface device. For instance, if past access history for an application indicates that a transmission operation of data size 4 Kilobytes over the network is always followed by a period of CPU-intensive operations without any accesses to the network, the energy management policy can then have the network card transition to a 'sleep' state at the end of every transmission operation of 4 Kilobytes (it would wake up back to its active state when there is a new packet to transmit or receive).

The *data mining* techniques involving classification based on decision trees were employed in this work for formulating and building the access patterns. The causal relationship among resource accesses and usage along with the time latencies separating them is captured by the probability

of their occurrence following a particular operation during application execution. The novel energy management policy presented in this work then determines an energy saving state for the network card based on the predicted pattern (one with the highest probability) of its accesses. Our proposed energy management policy decides on a power state for the network access card (that is, to keep it active or to switch to sleep state) following every operation involving access to it.

2 Related Work

Application-specific knowledge has vastly been exploited in the recent past to achieve optimum energy savings in mobile computing systems [6, 14]. These efforts employed application-specific information determined offline prior to their execution. In [13], an online approach to application characterization is presented. Information about application network behavior is obtained from the operating system and the application network access patterns are characterized using simple statistical techniques applied on past application behavior. The characterized application network access patterns are then mapped to a predefined set of application profiles that were determined offline. The network card was then switched accordingly between its power saving states based on the type of determined application profile. Our work which makes an effort to provide on-the-fly characterization of application behavior differs by using data mining concepts and directly applying the gained knowledge in predicting future network accesses for use in a dynamic energy management policy. We also aim to address the limitations of such an approach in interactive application environments (eg., multimedia server) where network access behavior may change dynamically over time with incoming service requests.

An effort related to ours was made in [3]. It employed pre-inserted hints from the application to obtain knowledge about its network usage and operations during run-time. We propose to achieve similar benefits, however with a more non-intrusive approach. Our work characterizes application behavior at run-time without ever having to modify the application code. There have been very few directed efforts in the past applying data mining and data mining techniques to address system level issues. Helmbold et., al [5] employed an ensemble data mining technique that utilized a voting mechanism to choose dynamically an optimal disk spin-down timeout factor. The algorithm presented selects an optimal timeout value from the weighted average of different possible timeouts values recorded over a period of time. This work however does not employ any information involving the current active set of applications in the target system.

Various other techniques employing prediction based models have been explored and employed in the context of several system level issues. The use of prediction models in operating systems and their effects were studied extensively in [8]. This work considers the specific problems of swapping pages in and out of caches and disk spin-down in computing systems in illustrating the benefits of employing online prediction algorithms.

Other existing energy management policies [2, 7, 11] consider the current active set of processes and/or their most recent invocations for the computation of optimal energy saving states. Attributes like system utilization, task deadlines are often the sole determining factors for the selection of a energy saving state (low speed, sleep state). But such attributes are only illustrative of the system usage and behavior at a given instant of time. This often leads to more aggressive and frequent energy management actions that achieve sub-optimal energy savings besides causing additional overhead on the system's resources.

3 Data Mining based Dynamic Energy Management

As discussed earlier, wireless network interface devices contribute significantly to the total energy consumption of mobile computing systems. The increasingly superior performance offered by modern network interfaces has accordingly resulted in a significant increase in their energy consumption. This evolving trend therefore demands the use of highly efficient and optimized energy management policies for such interfaces. Typical modern network cards are equipped with different built-in power

states (sleep, transmission, reception). Thus by making optimal use of the offered energy saving states, significant savings in the system-wide energy consumption can be achieved.

3.1 Energy Consumption Model

Conventional energy management techniques switch the network device to a low power state whenever there is no activity involving the device. That is, these *aggressive* ('greedy') mechanisms ensure that the network card stays at the active high power states only as long as they are transmitting or receiving data. Thus the energy consumed with an aggressive energy management policy is related to the number (n) and size of the data packets (s) being transmitted. The energy savings achieved with such a mechanism can hence be characterized by the following relations:

$$E(\text{aggressive})_{\text{consumed}} = \sum_{i=0}^n (E_{\text{pkt}} * s_i) + \sum_{i=0}^n E_{\text{switching}} \quad (1)$$

E_{pkt} describes the energy consumed in transmitting a packet of unit size assuming a typical fixed data transfer rate. The parameter $E_{\text{switching}}$ provides the additional energy overhead incurred during the transition between the power states (it captures the entire energy consumed in transitioning from the high power active state to its sleep state and then back again to its active state).

The energy savings achieved with any of the mechanisms can be normalized against the energy consumed by the device in the absence of any energy management mechanisms (denoted as E_{ON} which is the energy consumed when the device operates continually at its active power state during the corresponding duration of measurement).

$$E(\text{aggressive})_{\text{savings}} = E_{ON} - E(\text{aggressive})_{\text{consumed}} \quad (2)$$

Equations 1-2 show that the switching overhead also contributes significantly to the energy consumed in this model and thereby reduces the achievable energy savings during periods of high network accesses.

On the other hand, recently evolved energy management policies typically switch a wireless network interface device to the power saving sleep state based on a threshold value commonly referred to as *time-out* (TO). With such mechanisms, the network card is transitioned to its sleep state whenever there is no activity in the duration of the time-out value following a network access. A similar model describing the energy consumption of a network device employing a time-out based policy can be established. However, we need to consider in addition, the parameter of the number of packets (m) that are separated in time by a value greater than the chosen time-out factor (typically 100ms). This is due to the fact that the network card would remain at its active power state for the duration of the time-out value even in the absence of any network activity in that period. The model for the energy saved in a network card employing a simple time-out based mechanism is thus given as:

$$E(\text{timeout})_{\text{consumed}} = \sum_{i=0}^n (E_{\text{pkt}} * s_i) + \sum_{j=0}^m (E_{\text{switching}} + E_{TO}) + \sum_{k=0}^{n-m} (\frac{T_k}{TO} * E_{TO}) \quad (3)$$

$$E(\text{timeout})_{\text{savings}} = E_{ON} - E(\text{timeout})_{\text{consumed}} \quad (4)$$

The value denoted by E_{TO} represents the energy consumed when operating at the high power active state for the entire duration of the time-out period. The parameter T_k denotes the time latency between network accesses (corresponding to n-m packets) that occur within the timeout interval, ($T_{\text{current}} + TO$). It should be noted that the maximum energy savings achievable using such mechanisms are bound by the time spent in the time-out phase waiting for a network access.

Our approach by employing data mining techniques aims to achieve a compromise between both these approaches thereby maximizing the energy savings that can be achieved in such devices. The

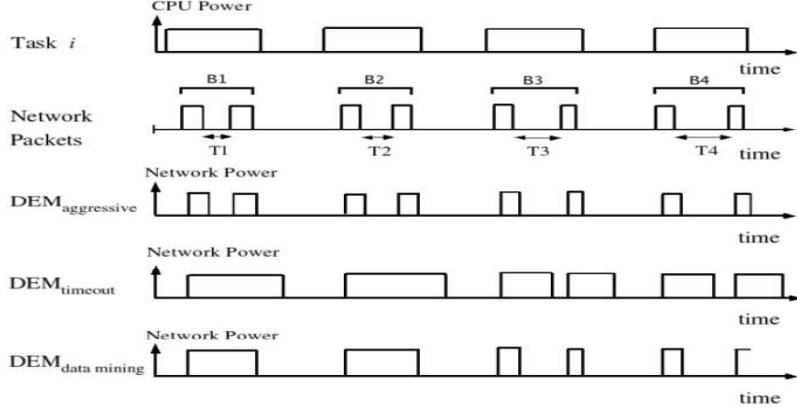


Fig. 1. Scenario illustrating the energy savings achieved in a network interface device for the various considered approaches.

presented mechanism in this work applies data mining concepts on a trace of application behavior over a recent past to formulate a model of the application’s behavioral and network access patterns. Using this model, the application’s future behavioral trends and activities can then be forecast from any given point of its execution cycle. This future knowledge is incorporated into the proposed energy management scheme as a set of rules that determine the current power state of the network device. Following every access to the network interface, our mechanism considers the next predicted network access and determines an energy state for the device. That is, the proposed mechanism allows the network card to stay at its active power state if it is determined that the energy overhead ($E_{switching}$) associated with switching the device between its active and sleep states is high relative to the energy that would be consumed staying at the active power state ($E_{TO'}$) until the next predicted network access. This amounts to determining an optimal time-out value TO' and deciding on a energy state for the device based on whether the next predicted access of this resource falls within this time-out value. An optimal time-out value is chosen such that,

$$E_{TO'} < E_{switching} \quad (5)$$

when network accesses occur within the time interval of TO' . Summarizing the proposed energy management scheme, we formulate the following rule set:

- a. If next predicted network access time is lower than the sum of the current network access time and TO' then stay at the active power state (ie., if $E_{switching} \geq E_{TO'}$).
- b. Else switch to sleep state until next access occurs.

Based on these formulated conditions, a model for our proposed energy management policy can be derived as follows:

$$E(TO')_{consumed} = \sum_{i=0}^n (E_{pkt} * s_i) + \sum_{j=0}^m E_{switching} + \sum_{k=0}^{n-m} \left(\frac{T_k}{TO'} * E_{TO'} \right) \quad (6)$$

From this model it can be ascertained that our mechanism would achieve higher energy savings compared to the other approaches. This is a result of our approach being able to cut down on the energy that would be consumed in the scenario where there are no network accesses in the time spent at the active state for duration of the time-out interval (ie., $\sum_{j=0}^m E_{TO}$ in Equation 3). Also this model reduces the number of transitions between the power states through the use of a new optimal time-out value (TO') thereby further maximizing the achievable energy savings.

3.2 Motivational Example

As an example, consider the scenario presented in Figure 1. Assume task i is a periodic multimedia application which processes media streams and sends them across the network to a requesting client.

The number of packets transmitted by this application over the network is related to the number and type of client requests. In this example, the task processes and transmits two packets of data as a burst every time it is invoked by the scheduler in the target system. The time latency involved between these packet transmissions is a function of the processing involved prior to their transmission which in turn is dependent on the type of client requests (eg., request for compressed vs non-compressed streams). The figure provides a comparison between the possible energy savings that can be achieved in a wireless network interface device with the existing aggressive, time-out based energy management mechanisms and our proposed scheme employing data mining techniques.

Let the latencies involved between packets in each of the bursts be such that $T1 < T2 < TO' < TO < T3 < T4$. It is evident that a *time-out based* policy would yield better energy savings over an aggressive mechanism for bursts B1 and B2. This results from the reduced switching overhead involved as the time-out based policy stays at its active state until the next packet arrives in each of these two bursts. However an aggressive mechanism would achieve higher energy savings for bursts B3 and B4 since the second packet in these bursts is generated beyond the time-out (TO) from the first packet transmission. As a result, the time-out policy would remain at its active power state even when there is no network access for the duration of the time-out from the first network access.

Our proposed approach, on the other hand, predicts the second network access in B3 and B4 to occur beyond the pre-determined optimal time-out (TO') thereby switching to the power saving sleep state as soon as the first network access is completed. At the same time, our approach would also overcome the switching overhead associated with an aggressive mechanism for bursts B1 and B2 by remaining at its active state for the optimal time-out duration (TO') since the first packet transmission. Thus our proposed approach would provide maximal achievable energy savings compared to the two existing widely adapted approaches.

4 Implementation Details

The platform used in this work was an Intel Pentium M based Gateway 450ROG Laptop. The operating system used was Fedora Core 3 running linux kernel 2.6.7.

4.1 System Resource Monitoring

The data containing information about the various operations and resource accesses made by all tasks invoked by the operating system since the beginning of the data acquisition process are collected and stored in the /proc interface in the kernel. The data is collected through the use of a custom built resource monitoring and logging tool named RESMONT. The tool is a loadable linux kernel module that traces accesses and operations performed on various resources by executing tasks. This tool is capable of gathering information about accesses to the resources of network card, system memory and disk besides logging exact task execution periods. Since this data collection tool is an architecture independent implementation, process traces can easily be obtained for different process execution environments thereby allowing for more generality in results and evaluations if required.

The data format required for extracting information about application behavior and its resource access patterns is identified as the following:

Timestamp, ProcessID, Resource, Operation, DataSize, Process Name

Timestamp captures the time at which the corresponding operation or resource access occurs. The time granularity for the logged data is in micro-seconds. *ProcessID and Process Name* provide information on the current task that is involved in the logged operation or resource access. The parameter *Resource* indicates whether resource currently accessed is that of Network, Disk or CPU. The value indicated by the *Operation* field denotes the type of operations performed such as Context Switch between tasks (CTXT), Read (READ), Write (WRTE), network packet enqueue (ENQU), Transmit

```
24,3681,mem,WRTE,23,bash
46,0000,cpu,CTXT,0,bash,mozilla-bin
159,3681,mem,READ,20,mozilla-bin
177,4395,net,ENQU,540,mozilla-bin
210,4395,net,TXEN,796,mozilla-bin
```

Fig. 2. A sample trace showing logged information about application behavior and its system resource usage.

End (TXEN), while *DataSize* indicates the size of the data involved in this current operation. The data collected is in the form of a contiguous trace of logged information about processes and their operations or resource accesses in their respective invocations (refer Figure 2).

The size of the trace files and the parameters logged can be specified and controlled at compile time of the monitoring tool.

4.2 Data Processing and Mining

The obtained data traces are then provided to filtering mechanisms that parse through the data performing formatting operations essential before use in a Data Mining environment.

The size of the data being involved during an operation or access is available as a continuous feature in the trace data. A simple discretization is performed on the values of the feature vector concerning size of operated data so as to achieve a fair degree of generalization in classification and thereby avoiding overfitting of training set data in the classifier rules during the learning phase. A holistic approach is employed in discretizing this feature vector.

Current efforts are still underway to have a data mining mechanism built into the operating system that is accessible to the monitoring and the proposed energy management mechanism that are implemented in the kernel. For this work, the offline data mining tool Weka [1] was used extensively in the application of all data mining techniques and mechanisms. The techniques of classification based on decision trees were employed. The cleaned and pre-processed data is thus applied as an input to the classifier tool based on a decision trees routine in Weka.

Classification Techniques. The generation of the resource access patterns for a particular application requires causal relationships to be established between the various accesses to resources made during the course of an application's execution. This can directly be mapped to a classification problem where given a particular operation or access to a resource as a feature vector, its corresponding class that would translate to the most probable subsequent operation is to be determined. That is the data applied to the classifier has the following characteristics: the feature vectors would consist of the given operation or access to a resource and the size of data used during such an operation while the class would be the operation or access to a resource that immediately follows the given operation in time. The assignment of an instance to a class variable in such mechanisms is made based on the highest probability of a given instance being assigned to that class. Thus, the probability used in assigning the class would be a reflection of the total number of instances when a particular operation or access to a resource (class) immediately succeeds a given operation or access (test instance).

The classifier technique based on decision trees was used for formulating the resource access patterns for a given application from the collected traces. The decision tree tool of REP-Trees in Weka was employed in formulating the resource access patterns for the considered applications that are common in the target execution environment. The REP-Tree procedure builds a decision tree using information gain as the splitting criterion, and uses reduced-error pruning for pruning. This procedure is also characterized by lower computational overheads compared to other decision tree based classification methods as a result of its efficient pruning mechanism. The classification technique of decision trees was employed so as to capture the underlying dependencies between the feature vectors and thereby providing accurate models.

5 Experimental Evaluations

By applying predictions of future accesses to the network interface device using the characterized application behavior model, our proposed energy management policy is shown to achieve optimal achievable energy savings. For the experiments in this work, a few common wireless network interface cards were considered as the resource of interest in a mobile computing system (Gateway 450ROG Laptop). It was determined experimentally that the energy consumed at the high power active state is as high as 95% of that consumed at the power saving sleep state (eg., in the Orinoco-Gold card, the power consumed in the active state is 180mA while the power consumed in the sleep state is only 9mA). The switching overheads involved in transitions between the available power states play a significant role in the determination of the break-even time for a network interface card. The break-even time ($T_{breakeven}$) is defined as the amount of time that a resource must remain in the sleep state before less energy is consumed than staying in its active state [10]. It therefore represents the time beyond which remaining at the active state without any network accesses would consume more energy than switching to the sleep state until an access eventually occurs. This value is computed as,

$$T_{breakeven} = \max\left\{\frac{E_0 - (P_{sleep}) * t_0}{(Power_{idle} - Power_{sleep})}, t_0\right\} \quad (7)$$

where $t_0 = T_{switch}(active - sleep) + T_{switch}(sleep - active)$ and $E_0 = (T_{switch}(sleep - active) + T_{switch}(active - sleep)) * Power_{switch}$.

From our experiments, it was determined that the breakeven values typically range between 3-4ms for commonly available wireless network cards. However, it should also be noted that most of the network cards come with built-in power saving mechanisms (named Power Saving Mode, PSM) that provide their own energy management mechanisms. It was determined in [3] that the switching overheads involved with transitions between the PSM and the continually aware mode (with no power management) is in the order of 200ms for typical network cards. Also this related work shows that employing PSM does not always provide significant benefits to the end user. It proposes switching between the PSM and the continually aware mode (CAM) depending on the application network access behavior. Thus when switching between these two power modes, the break even time is in the range of 200-300 milli-seconds.

All traces concerning operations and access to this resource were extracted from the obtained log files and used for analysis. The application analyzed and presented in this work is a command line based image distribution tool named *PPM-Share*. This image distribution tool provides an environment for sharing and distribution of different image formats over the network. It is based on a client-server model and provides for distribution and sharing of images over a variety of formats such as raw, compressed or scaled. This application was chosen for all our experimental evaluations as a result of its close resemblance to a dynamic application environment whose resource access behavioral patterns vary over time depending on external parameters such as incoming client requests. The application behavior model was formulated with the trace files that were collected with the above application running as a isolated single instance in the execution platform.

Methodology. The algorithm employed in our experimental simulations comparing the energy savings achievable with the proposed mechanism against existing energy management approaches for the network card is presented here (Algorithm 1). The monitoring tool was used to monitor and collect traces of all operations and resource accesses performed by the considered application for the duration of its execution. The collected traces over an interval of 20 seconds was applied to the REP-Tree classification procedure in Weka. Using the classification rules devised by this mechanism, a Network Resource Access Pattern (NRAP) is formulated for the chosen application. The formulated NRAP representation modelling recent behavior and network accesses is then employed in making fair predictions of future network accesses by this application.

Our proposed Energy Management Mechanism ($DEM_{DataMining}$) uses a dynamic optimal time-out (TO') value in transitioning the network device to its power saving sleep state. The value of TO' is computed as the earliest of the break-even time ($T_{breakeven}$) and the next predicted network access of the application so as to avoid unwanted transitions due to predicted. In order to avoid frequent transitions between the power states due to variations between the predicted and actual future access times that are too low, a prediction window (w) is applied to the predicted future network access. Our mechanism thus considers the predicted network access to occur in the interval,

$$t_{predicted} = t_{next} * w \quad (8)$$

Currently, a holistic value of 2 was employed for the prediction window (w). Our ongoing work aims to devise a simple procedure for dynamically computing an optimal window threshold that would further reduce this cost of variations in predicted and actual times in our approach.

Algorithm 1 Pseudocode for $DEM_{data mining}$

Require: Execution and resource access traces of the analyzed application.

- 1: At the end of each *Network Access*
 - 2: $DEM_{data mining}()$

 - 3: **Monitor():**
 - 4: trace resource accesses and execution of the target application in the target system.
 - 5: **for** every 20 seconds **do**
 - 6: Data Mining()

 - 7: **Data Mining():**
 - 8: Run REP-Tree algorithm on the traces collected over the last 20 seconds.
 - 9: Store the classification rules (NRAP) obtained for the accesses concerning the network card.

 - 10: $DEM_{data mining}()$:
 - 11: From the mined NRAP model predict the time of next network access (t_{next}).
 - 12: Determine the value of the TO' as $\min(T_{breakeven}, 2 * t_{next})$.
 - 13: Allow network to stay at its active state for the duration of TO' following the current network access.
 - 14: **if** no network access occurs in this time-out interval **then**
 - 15: switch network card to *sleep* state.
 - 16: **else**
 - 17: goto 1.
-

5.1 Result Analysis and Discussion

An evaluation of the energy savings achieved by the proposed mechanism employing data mining are presented here. Figure 3 compares the energy savings achieved by the existing mechanisms based on aggressive ($DEM_{aggressive}$) and time-out policies ($DEM_{timeout}$) against our proposed mechanism ($DEM_{data mining}$). The time-out based mechanism compared in this section is assumed to have a typical time-out value (TO) of 100milli-seconds.

The execution and resource accesses of the server application, ppm-share, were monitored and traced during its service of incoming requests from a remote client. Requests typically involved transfer of images that were cropped, scaled, encoded by the server application according to the parameters passed from the requests. The trace data was then applied as an input set once every 20 seconds to the data mining mechanism of decision tree formulation using REP-tree algorithm available in the Weka interface. The size of the trace data collected over a interval of 20 seconds for the above application ranged between 12 - 12.5 Kilobytes consisting of 550-600 trace lines in Weka compatible

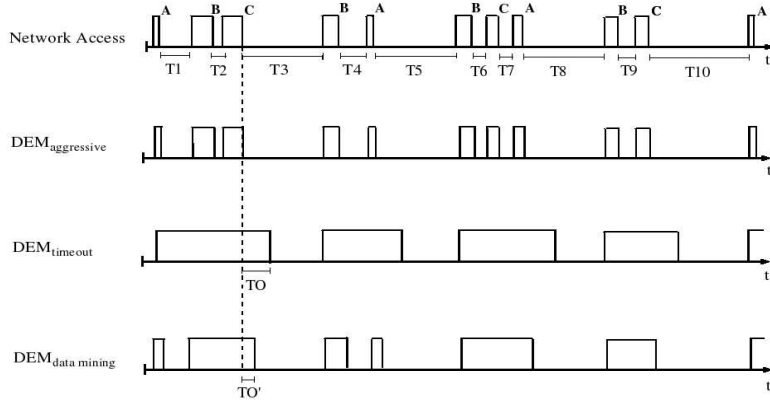


Fig. 3. Comparison of the Energy Savings achieved by our proposed mechanism against existing approaches.

format. A representation of the decision tree classification provided by this mechanism is shown in Figure 4.

Using the formulated decision trees, future network accesses were predicted which were then employed in our mechanism to optimize energy savings. Figure 3 shows a snapshot of accesses made to the network card by the application represented as Network Resource Access Pattern (NRAP). Predictions about the next network access by the application were made at the end of each of its network operations. Each network operation performed by the considered application were distinguished by the type (ENQU, TXEN) and size of data involved in the operation. A typical network access may comprise of many such closely spaced network operations (eg., ENQU operations followed by TXEN operations). The variables A, B and C in Figure 3 represent the last network operation performed before the end of a network access. They correspond respectively to the operations of TXEN involving a data size of less than 200 bytes, less than 1000 bytes and greater than 1000 bytes. The decision tree in Figure 4 describes the next predicted operation (leaf nodes) that would follow each of these above given operations. The average time latency (in milli-seconds) expected before the next predicted operation occurs is shown as weights on the tree edges. Assume a typical break even time, $T_{breakeven}$ as 10ms computed by including a correction factor to the theoretically determined value from Equation 7. A pessimistic correction factor is used so as to avoid frequent transitions between the power states that would reduce the achievable energy savings.

It can be seen from the graph that our approach employing data mining techniques is able to achieve better energy savings by maximizing the time spent by the network card in the power saving speed state. For instance, at the end of the first operation of type C (TXEN of size greater than 1000 bytes), the mechanism uses the formulated decision tree (in figure 4) to predict the next network operation of ENQU involving size less than or equal 1000 to occur within the next 2.75 milli-seconds (t_{next}). The $t_{predicted}$ for the next access would thus be computed as 5.5 milli-seconds. The optimum TO' value computed by our mechanism for this case would therefore be 5.5 milli-seconds ($\max(10, 5.5)$ ms) and the network card is allowed to remain in its active state for this duration following the current access. The actual next access to the network occurs ($T2=1.6$ ms) within this time-out value and as a result of the network card remaining at its active state, our mechanism is able to achieve greater energy savings comparatively. A time-out mechanism in this case would stay at the active state for a longer duration as result of the larger timeout value thereby consuming more energy. A similar scenario occurs following the sixth network access ($T6=2.8$ ms) involving a TXEN operation of data size less than 1000 bytes. Our mechanism achieves further energy savings by having the network card stay at its active state at the end of the seventh and ninth network accesses, of type C and B respectively, following which network accesses occur ($T7=3$ ms, $T9=6.4$ ms) in their corresponding optimal time-out intervals ($TO' = 6$ and 10 ms respectively). At all other times ($T1=12.8$ ms, $T3=345$ ms, $T4=38.5$ ms, $T5=330$ ms, $T8=347$ ms, $T10=316.7$ ms), the network card is transitioned to a sleep state when there is no network access in the optimal time-out duration following that current network operation. Also in comparison to a time-out based approach, the network card in our approach spends less time in

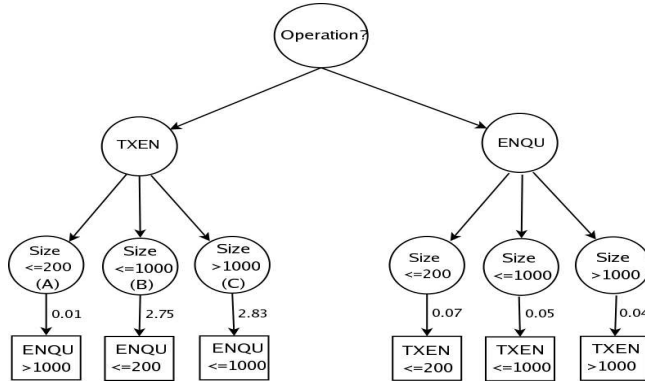


Fig. 4. Formulated Network Resource Access Patterns (NRAP) for the analyzed application (ppm-share).

the active state waiting for a future network access (evident from Figure 4 showing the duration of TO and TO') comparing resulting in higher energy savings as. The optimal time-out employed in our approach also brings down the number of transitions between the power states that incur higher energy overheads in an aggressive energy management policy.

The energy savings achieved by our proposed mechanism can further be increased by switching between the PSP and CAM modes that incur higher overheads (order of hundreds of milli-seconds) in switching between them in a simple time-out based policy ($DEM_{timeout}$).

5.2 Overhead Considerations

One of the significant factors in the design of our energy management approach was the overhead involved and their effects on the overall benefits achieved. The overhead of the data collection module was found to be very negligible as result of its execution from the the kernel space where all relevant information about the execution environment is stored. The significant overhead introduced by our mechanism resulted from the data mining procedure.

As described in the previous section, experimental simulations involved executing the data mining procedure every 20 seconds on a trace set consisting of about 600 lines after processing and conversion to Weka compatible formats. The REP-Tree procedure was used with all known computationally expensive features such as pruning turned off. The overhead contributed by the data mining procedure employing the REP-tree algorithm for decision tree formulation on this data set is determined to be around 180-200 milli-seconds. This results in a overall overhead of 1% as a result of the data mining procedure being run every 20 seconds. Our future work would further aim to bring down the associated overheads by providing an online in-built interface in the operating system of the target system for data mining and analysis.

As part of our current on-going work, similar on-line approaches employing data mining mechanisms are being considered and developed for addressing issues concerning energy minimization for the entire system combining the available processor, network and disk power saving states and mechanisms. The resulting overhead for the data mining mechanism would thus be shared by these mechanisms thereby resulting in significantly lower overheads compared to the achievable system-wide energy savings.

6 Conclusion and Future Work

A novel energy management policy maximizing the achievable energy savings employing Resource Access Patterns (RAP) characterized using data mining approaches was presented in this work. The data mining approach by providing a online non-intrusive mechanism for application behavior characterization was able to achieve fair predictions on future network accesses of the application. An optimal time-out value based on the next predicted network access was then employed to decide on

a power saving state for the network card following every access to it. As a result of this dynamically computed optimal time-out value, our mechanism was able to achieve better energy savings compared to aggressive and static time-out based approaches.

As part of our on-going work, we intend to analyze the effects of mis-predictions on the achievable energy savings and explore ways to achieve significant benefits even in the presence of tolerable mis-predictions. Another issue addressed in our on-going efforts is that of building access pattern models covering different application actions and resource accesses within a considered time interval. Such models would be able to provide energy savings in the presence of several concurrently executing applications.

Our future work would focus on improving the accuracy and the interface offered by data mining mechanisms. One particular course of action would involve incorporating a dynamic self-adaptive approach in our existing mechanism that would switch between different data mining techniques as well as various accuracy enhancing features such as pruning and cross validation, depending on their accuracy, overhead and system resource and energy availability. Another interesting system level optimization problem concerning effective task scheduling is also being considered in our efforts for developing mechanisms employing data mining techniques.

References

1. Department of Computer Science, University of Waikato, Hamilton, New Zealand, <http://www.cs.waikato.ac.nz/~ml/index.html>.
2. N. AbouGhazaleh, D. Moss, B. Childers, R. Melhem, and M. Craven. Collaborative operating system and compiler power management for real-time applications. In *RTAS '03: Proceedings of the The 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, Washington, DC, USA.
3. M. Anand, E. B. Nightingale, and J. Flinn. Self-tuning wireless network power management. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom)*, pages 176–189, New York, NY, USA, 2003. ACM Press.
4. L. M. Feeney. Investigating the energy consumption of an iee 802.11 network interface. In *Technical Report SICS T9911*.
5. D. P. Helmbold, D. D. E. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *Proceedings of the Second Annual ACM International Conference on Mobile Computing and Networking, 1996*, pages 130–142, 1996.
6. C. Hsu and U. Kremer. Compiler-directed dynamic voltage scaling for memory-bound applications. In *C.-H. Hsu and U. Kremer. Compiler-directed dynamic voltage scaling for memory-bound applications. Technical Report DCS-TR-498, Department of Computer Science, Rutgers University, August 2002.*, 2002.
7. R. Jejurikar and R. Gupta. Dynamic Voltage Scaling for Systemwide Energy Minimization in Real-time Embedded Systems. In *Proc. of the 2004 International Symposium on Low Power Electronics and Design*, pages 78–81, New York, NY, USA, 2004. ACM Press.
8. P. Krishnan. Online prediction algorithms for databases and operating systems, technical report cs-95-24. 1995.
9. K. Li, R. Kumpf, P. Horton, and T. E. Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter Conference*, pages 279–291, 1994.
10. Y.-H. Lu and G. D. Micheli. Comparing system-level power management policies. *IEEE Design and Test of Computers*, 18(2):10–19, March 2001.
11. C. Poellabauer and K. Schwan. Energy-aware traffic shaping for wireless real-time applications. In *IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, May 2004*.
12. A. A. P. G. T. Simunic, L. Benini and G. D. Micheli. Dynamic Voltage Scaling and Power Management for Portable Systems. In *Proc. of Design Automation Conference, 2001*.
13. A. Wei, M. Faerber, and F. Bellosa. Application characterization for wireless network power management. In *Proceedings of the International Conference on Architecture of Computing Systems (ARCS'04), Augsburg, March 2004*.
14. F. Xie, M. Martonosi, and S. Malik. Compile-time Dynamic Voltage Scaling Settings: Opportunities and Limits. In *Proc. of the ACM SIGPLAN Conference on Programming Languages Design and Implementation (PLDI'03)*, June 2003.