POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Energy Management for IoT

Lab 1 - Report

Dynamic power management

Dario Castagneri
277967

A. Y. 2020-2021

Table of contents

Introduction

The main goal of the experience is the analysis on how power management policies can impact on power consumption in different workloads scenarios. Three different kinds of policies have been analysed: two timeout based policies, one with a single threshold, the other with a double threshold and a history based policy. The workloads are generated according to different distributions of active and idle periods. The power management policies and their energy statistics are simulated and computed by using the provided DPM simulator, MATLAB and bash scripts. All the bash scripts presented in the following are wrapped in a single script and they can be executed through *scriptLaunch.sh*.

Workload profile generation

The uncorrelated workloads profiles

The requested workloads have been generated using the *generating_workloads.m* MATLAB script. The active times are always referred to the same type of uniform distribution, while the idle times are generated according to different models: two types of uniform distribution to simulate different utilization profiles, a normal, an exponential and a trimodal distribution. The graphical representations of each distribution profile are shown in *Figures [1-6].*

Active periods All the active periods have been determined using the same model: a uniform distribution between a = 1 μs and b = 500 μs. (*Figure 1*)
Idle periods - High and Low utilization The first idle period type refers to a high utilization profile. It is generated from a uniform distribution between a = 1 μs and b = 100 μs. (*Figure 2*). Instead, to model a low utilization profile that requires to generate more idle periods with respect to the active ones, a uniform distribution between a = 1 μs and b = 400 μs has been used. (*Figure 3*)
Idle periods - Normal distribution In this case the idle periods refer to a normal distribution with a mean of 50 μs and a standard deviation of 20 μs. (*Figure 4*)
Idle periods - Exponential distribution In this case the idle periods refer to an exponential distribution with a mean of 50 μs. (*Figure 5*)
Idle periods - Tri-modal distribution The tri-modal distribution has been generated as a combination of three normal distributions with means 50 μs, 100 μs and 150 μs. The choice for the distribution is made according to an integer value randomly chosen between 1 and 3. (*Figure 6*)

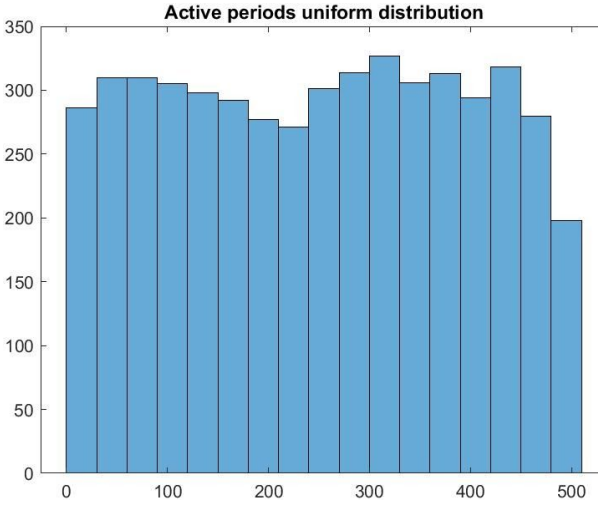


Figure 1

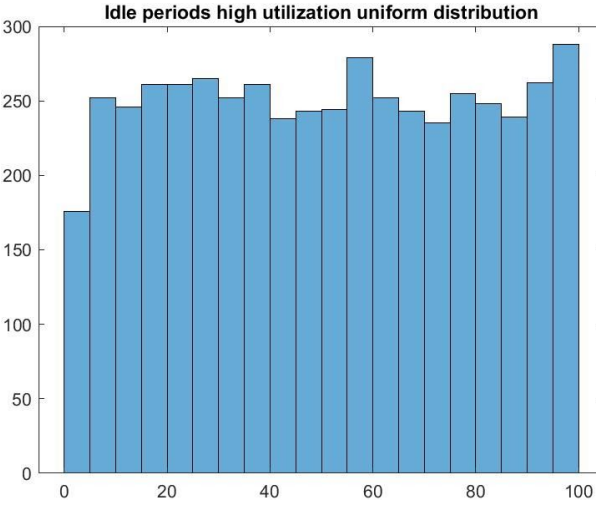


Figure 2

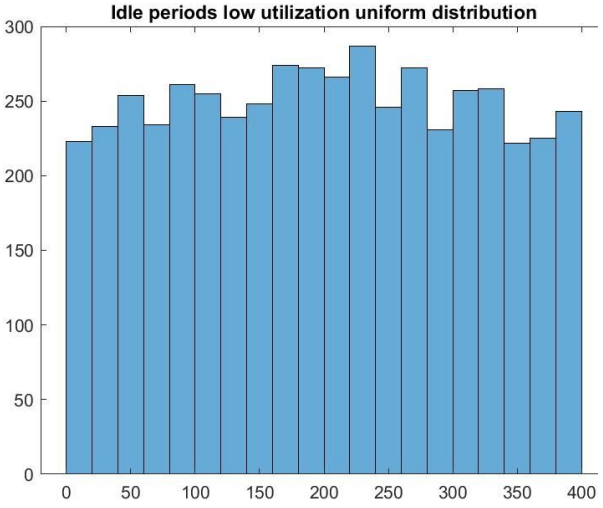


Figure 3

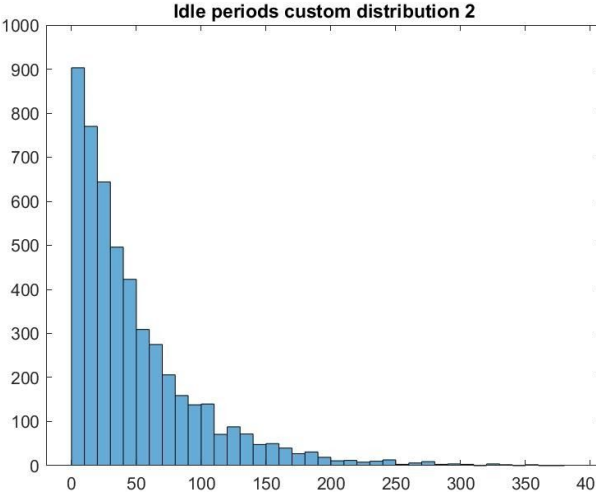


Figure 4

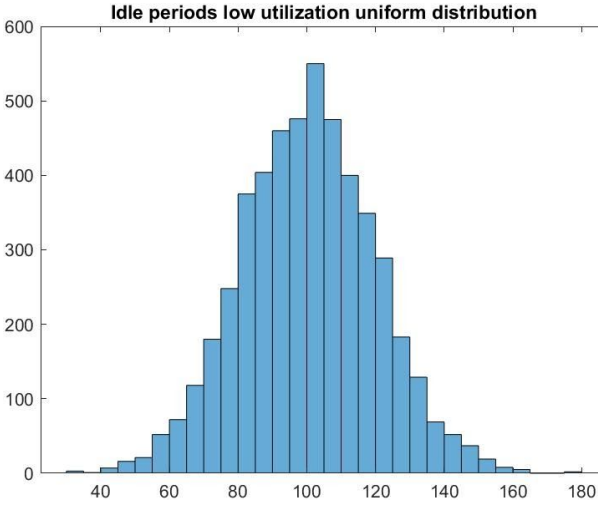


Figure 5

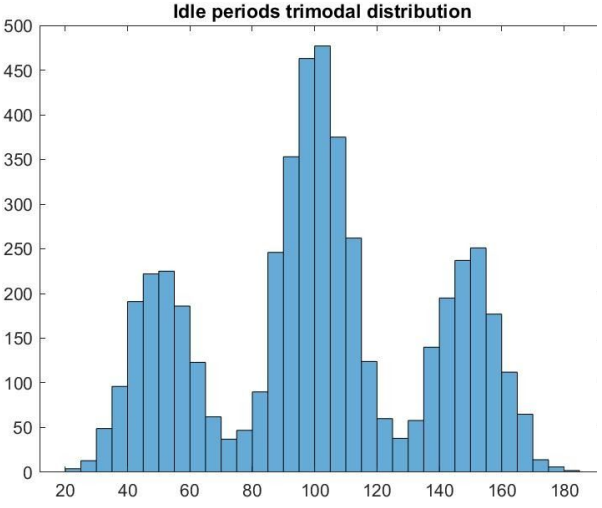


Figure 6

Custom workloads analysis The given custom workloads have been analysed using the MATLAB script *checking_given_workloads.m*. The script, once the workload files have been loaded, extracts the active and the idle values and generates their corresponding distributions. The graphical representations are shown in *Figure 7*, *Figure 8* and *Figure 9*, *Figure 10,* for the two workloads respectively.

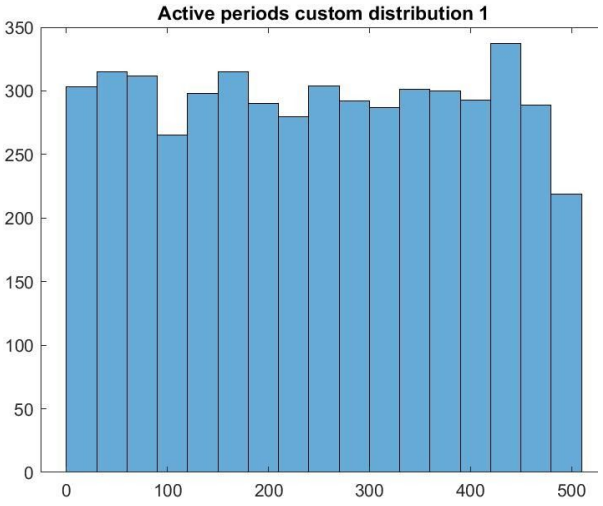


Figure 7

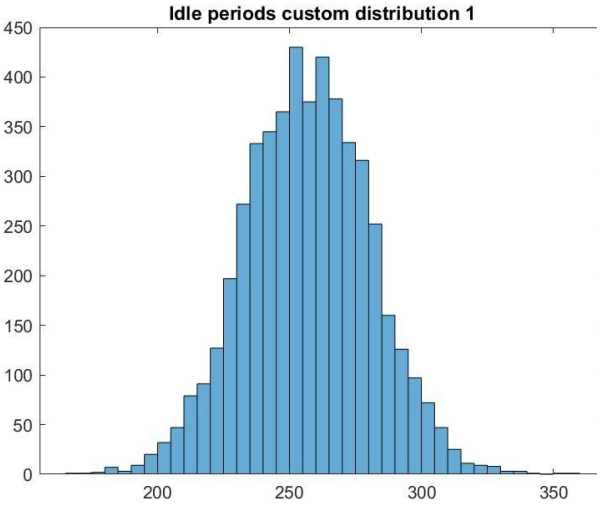


Figure 8

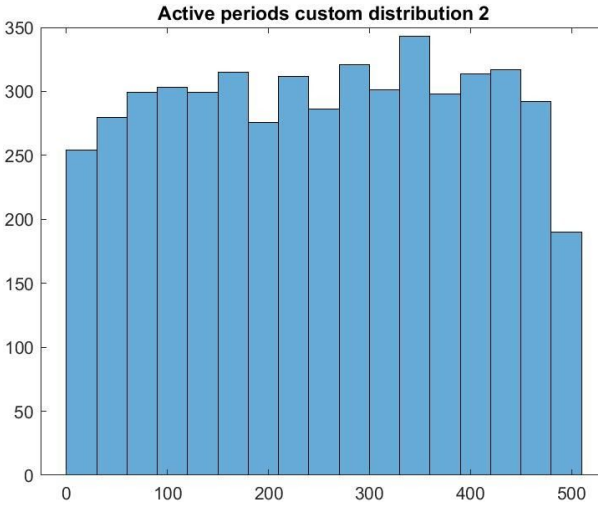


Figure 9

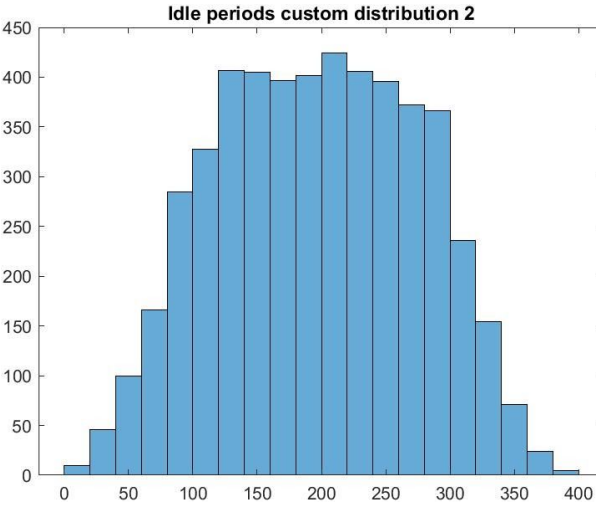


Figure 10

As we can see from the graphical distributions, for both the workloads, the active periods refer to a uniform distribution between 0 μs and 500 μs. Meanwhile, the idle periods have been generated from two normal distributions with means close to 260 μs and 200 μs.

The timeout policy

Introduction

In this section, a timeout policy is applied to two different power state machines. In the first PSM, transitions can happen to and from *idle* state only. In the second, transitions to and from *sleep* and *idle* states are also possible. The two PSM are shown in *Figure 11* and *Figure 12* respectively. For both the scenarios, different conditions are applied by tuning the timeout parameters that characterise each PSM low power state. In particular, a timeout threshold (*Tto_Idle*) whenever the idle state is present (*Scenario 1* and *Scenario 2*) and an additional timeout threshold (*Tto_Sleep*), whenever the *sleep* low power state is used (*Scenario 2*).

Scenario I - Single low power state

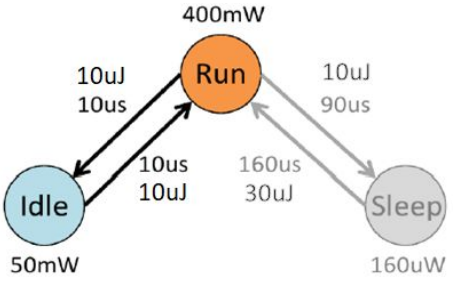


Figure 11

The first scenario PSM has a single low power state (*Figure 11*). In a timeout policy, the system remains in the active state until its computations are required. It may enter the idle state only if reaching the low power state, considering non-ideal transitions, has some advantages than remaining in the active state. The choice, performed during a period known as timeout time, is based on the evaluation of the length of the actual idleness. When the timeout expires, the evaluated idle time is compared with the break-even time. If the idle time is greater, it is convenient for the system to move to the low power state. If not, the system remains in the active state. The break-even time is computed analytically according to two formulas, depending on the PSM parameters:

$$Tbe = \begin{cases} Ttr + Ttr\frac{Ptr-Pon}{Pon-Poff} & \text{when } Ptr > Pon \\ Ttr & \text{when } Ptr \leq Pon \end{cases}$$

In order to understand which formula must be used, the power required to move the system from the *run* state to the *idle* state (and vice versa) is computed:

$$Ttr_{idle} = Ttr_{(Run->Idle)} + Ttr_{(Idle->Run)} = 10\mu S + 10\mu S = 20\mu S$$

$$Ptr = \frac{Etr}{Ttr} = \frac{Etr_{(Run->Idle)} + Etr_{(Idle->Run)}}{Ttr_{(Run->Idle)} + Ttr_{(Idle->Run)}} = \frac{20\mu j}{20\mu s} = 1W$$

Given $P_{on} = P_{Run} = 400mW$ and $P_{off} = P_{Idle} = 50mW$.

Being $P_{tr} > P_{on}$, the break even time is then computed according to the first formula:

$$Tbe = Ttr + Ttr\frac{Ptr - Pon}{Pon - Poff} = 20\mu s + 20\mu s\frac{1W - 0.4W}{0.4W - 0.05W} = 54.3\mu s$$

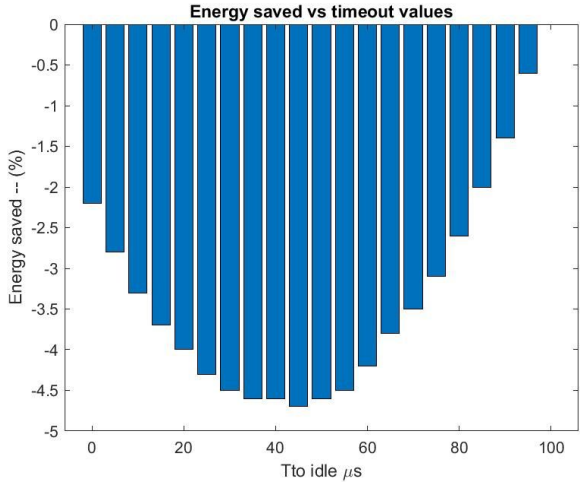
Generally, as the timeout value increases, it is expected that predictions become safer, with less performance penalty, but less efficient in terms of power consumption.

Scripts for automatic power analysis

In order to automate the generation of power statistics related to different workload profiles and different timeout values, the *scriptStandardVersion.sh* bash script has been used. The initial *idle* timeout is set to 0 for all the profiles and then incremented by 5 µs until the upper limit is reached. The upper limit value has been decided independently for each type of workload, according to the presence of significant values in the power report. Once the power statistics have been generated, the script extracts the percentage of energy saved. By using *energyVsTimeout_standard_version.m* MATLAB script, graphical representations of how the power consumptions change with respect to different timeout values are generated. A similar approach for the custom workloads is used in *scriptStandardVersionCustomWLs.sh, energyVStimeoutCustom1.m* and *energyVStimeoutCustom2.m* scripts, to respectively compute energy statistics and the associated graphical representations.

Experimental results

Workload Profile 1 - Idle times from a uniform high utilization distribution



Active time in profile = 1.250908s
Idle time in profile = 0.251860s
Total time = 1.502768s

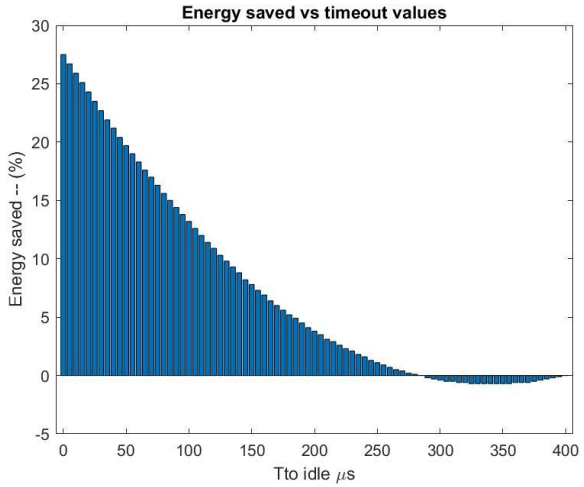
Tto_Idle at 20 µs:
Timeout waiting time = 0.094943s
Total time in state Run = 1.345851s
Total time in state Idle = 0.156917s
Total time in state Sleep = 0.000000s

Figure 12

The *dpm_simulator* output confirmes the high utilization profile. In fact, the total active time is approximately 5 times higher than the total idle time. By looking at other *dpm_simulator* statistics, specifically those related to the simulation with the timeout set to 20 µs, it is possible to appreciate additional information about the implemented policy. As an example, the *timeout waiting time* refers to the additional time spent by the

system while being in the *run* state. During this period, the system verifies if the incoming idle period will exceed the timeout threshold of 20 μs. From *Figure 12*, it is evident that the presence of the implemented policy does not show any benefits for the high utilization profile. In fact, the power consumption is negative for every timeout value. This means that the DPM makes the system consume more power. As an example, when the timeout is set to 45 μs, the simulator shows a 4.7% increment in power consumption, the highest amount of wasted power. The reason is to be found in the fact that the system is waiting in the *run* state for an additional time of 45 μs before deciding if it is convenient to move to the *idle* state. During this amount of time, being in the *run* state, the system consumes a higher amount of energy. For this specific case, the remaining time in which the system remains in the *idle* state is 100 μs - 45μs = 55 μs. In addition, the increment in power consumption for timeout values up to 45 μs can be explained by the fact that for those values, the remaining portion of the overall idle period, according to which the system decides whether to move or not to the *idle* state, will be smaller than that of the best scenario (100 μs - 45 μs = 55 μs). Considering the definition of break even time, the remaining portion will be likely smaller than the break even time and for this reason, the system will remain in the *run* state consuming a higher amount of energy. On the other hand, the decrease in the additional power requests for timeout values greater than 45 μs, is explained by a higher timeout value that reduces the PSM number of opportunities to make a transition into the *idle* state.

Workload Profile 2 - Idle times from a uniform low utilization distribution



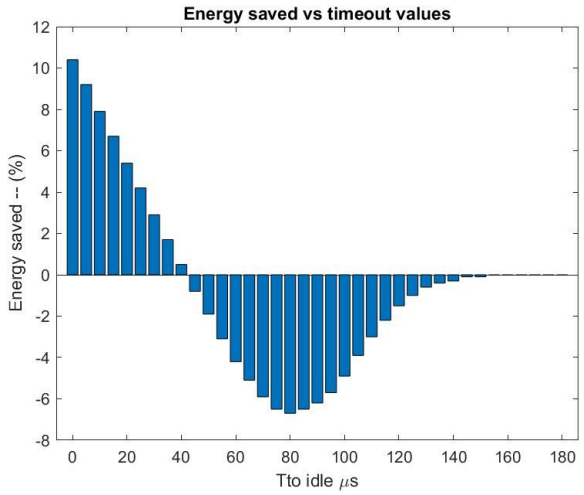
Active time in profile = 1.250908s
Idle time in profile = 0.998080s
Total time = 2.248988s

Figure 13

The data from the simulator confirms that, since the ratio between the total active and idle time is closer to 1, the distribution is referred to a low utilization profile. Because of that, as it is evident from *Figure 13*, the timeout policy has a positive impact on energy savings, especially with low timeout thresholds. The main reason for such a difference with respect to the previous workload profile is that the system has now more time to enter the *idle* state and benefit in higher energy efficiency. Another interesting

observation can be made. For a timeout value equal or greater than 285 µs, the system energy consumption starts to increase. The worst increment is achieved at 340 µs. The reason is that, as for the previous profile, for values greater than 285 µs, there is a higher probability that the remaining time in which the system can stay in the *idle* state is lower than the break even time.

Workload Profile 3 - Idle times from a normal distribution

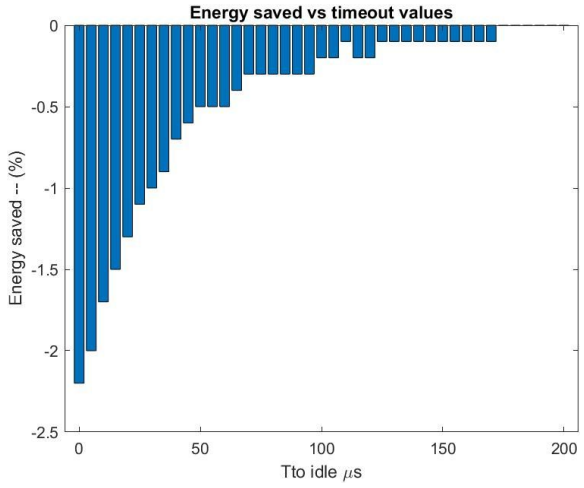


Active time in profile = 1.250908s
Idle time in profile = 0.499651s
Total time = 1.750559s

Figure 14

As in the first workload profile, the idle times are too short with respect to the active ones. For this reason, the timeout policy does not show any overall benefits in terms of power savings, especially for timeout greater than 45 µs.

Workload Profile 4 - Idle times from an exponential distribution

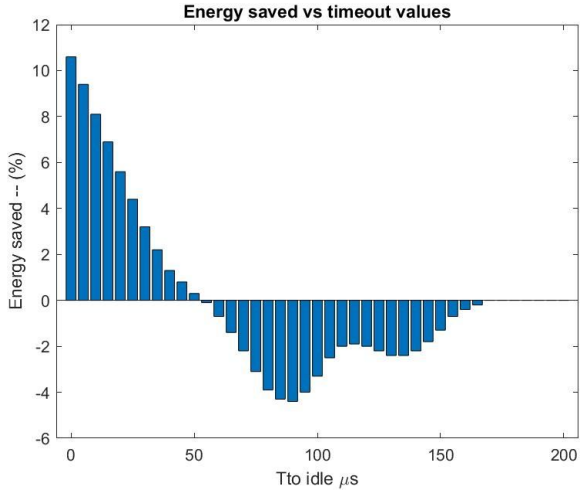


Active time in profile = 1.250908s
Idle time in profile = 0.244058s
Total time = 1.494966s

Figure 15

Same observations as the first and previous profiles can be made. The ratio between the active and idle times is too high. For this reason, the timeout policy does not show any overall benefits in terms of power savings. The energy consumption is higher with respect to the same system without a DPM policy.

Workload Profile 5 - Idle times from a trimodal distribution

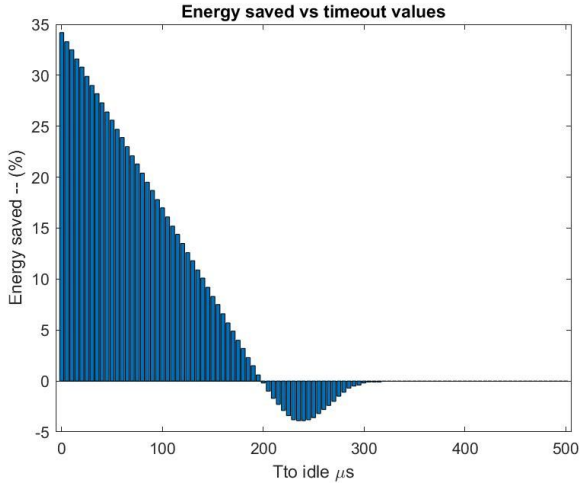


Active time in profile = 1.250908s
Idle time in profile = 0.503344s
Total time = 1.754252s

Figure 16

The results for low timeout values are very similar with the results from the third workload profile. Instead, for higher timeout values, there is an overall increment in terms of power consumption, but not as in the third profile. The reason is probably related to the added variability of a trimodal distribution with respect to a single normal distribution profile.

Custom Profile 1

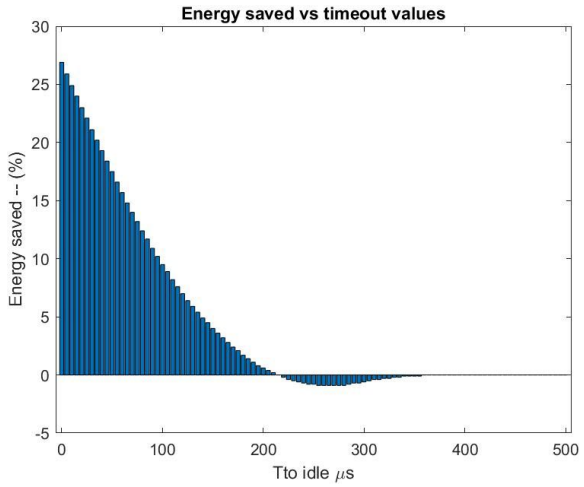


Active time in profile = 1.253796s
Idle time in profile = 1.282248s
Total time = 2.536044s

Figure 17

For the *custom profile 1*, the active and idle times are very similar. In fact, the timeout policy shows a good amount of energy saved, especially for low timeout values. The increment of power consumption for timeout greater than 200 μs can be explained as in the first profile.

Custom Profile 2



Active time in profile = 1.269283s
Idle time in profile = 0.982149s
Total time = 2.251432s

Figure 18

Similar considerations can be made for the second custom workload which appears to have a lower impact on both positive and negative energy savings.

Scenario II - Double low power states

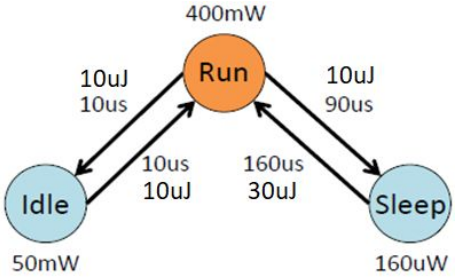


Figure 19

The PSM has now an additional low power state, consequently an additional timeout threshold for the *sleep* state is added. Assuming its timeout threshold is greater than the *idle* one, whenever the system may enter a low power state, the DPM checks the actual inactive state. If the *idle* state threshold expires and the system is still inactive, there will be a transition to the *idle* state. Moreover, if the *sleep* state threshold expires as well, the system will move to the *sleep* state. Whenever it is requested, the system goes back to the *run* state. In order to support the extended PSM, the *dpm_simulator* has been further modified. At each instant, the current *idle* time is compared with the *idle* timeout threshold. If greater, the system reaches the *idle* state. If not, the system remains in the *run* state. Following the same principle, the system may or may not reach the *sleep* state.

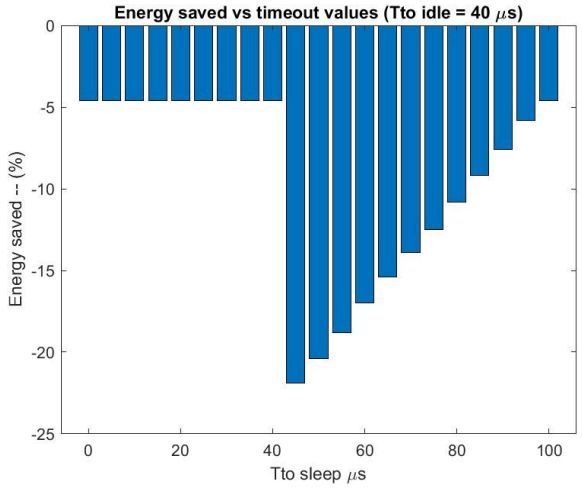
Scripts for automatic power analysis

To automate the simulation process, two bash scripts have been used: *scriptLaunchExtendedTimeoutSleep.sh* and *scriptExtendedTimeoutSleep.sh*. The first script launches the second with different *Tto_Idle* values according to the workload profiles.

Different values for the *sleep* timeout, increased by 5 µs each time, are set by the script. At the end, the script gathers all the power statistics and MATLAB is used to make graphical representations of power savings and the average time spent in each state, for each workload. Because of the high number of simulations, only a subset of all the possible cases are shown in the following. Specifically, for each workload, a single graph associated to a specific *idle* timeout threshold is provided.

Experimental results

Workload Profile 1 - Idle times from a uniform high utilization distribution



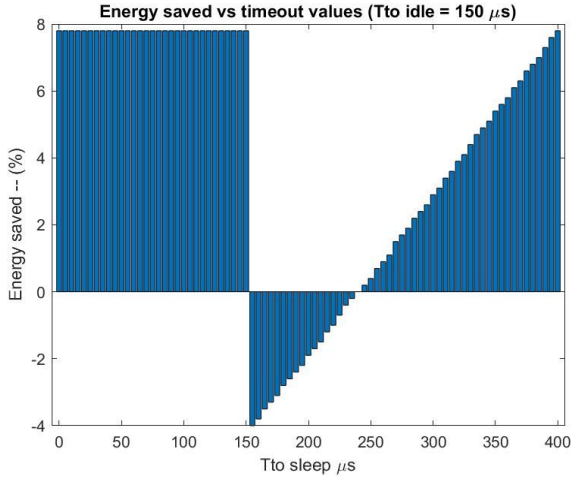
Average time in states:

Run: 1.412924s
Idle: 0.068791s
Sleep: 0.021052s

Figure 20

As for the previous PSM, the high utilization profile does not benefit from this policy. An increase in power consumption is shared by all *sleep* timeout values. Overall, the energy drawbacks are even worse: in the first PSM, the worst value was a power increment of 4.7%, here, values above 20% are reached. In addition, only when the *sleep* timeout is greater than the *idle* threshold, the system can reach the *sleep* state. This behavior is evident for the simulations with a *sleep* timeout above 45 µs.

Workload Profile 2 - Idle times from a uniform low utilization distribution

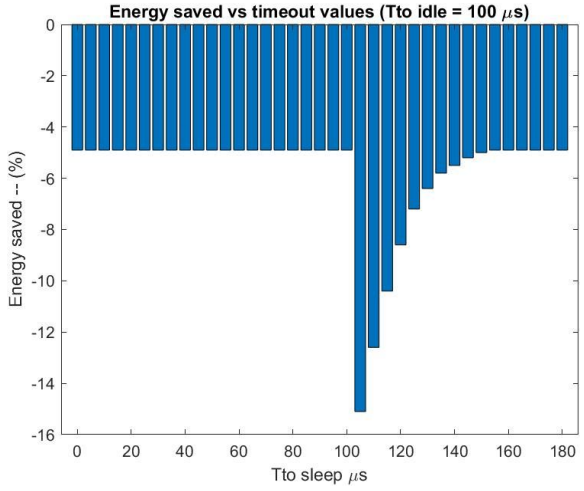


Average time in states:

Run: 1.822471s
Idle: 0.317151s
Sleep: 0.109366s

Figure 21

Workload Profile 3 - Idle times from a normal distribution

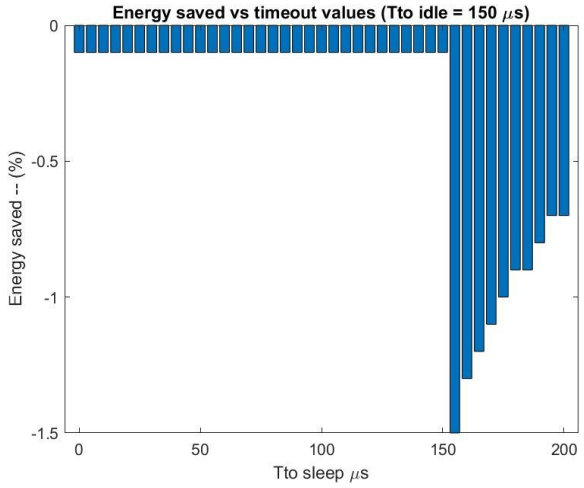


Average time in states:

Run: 1.568775s
Idle: 0.142843s
Sleep: 0.038942s

Figure 22

Workload Profile 4 - Idle times from an exponential distribution

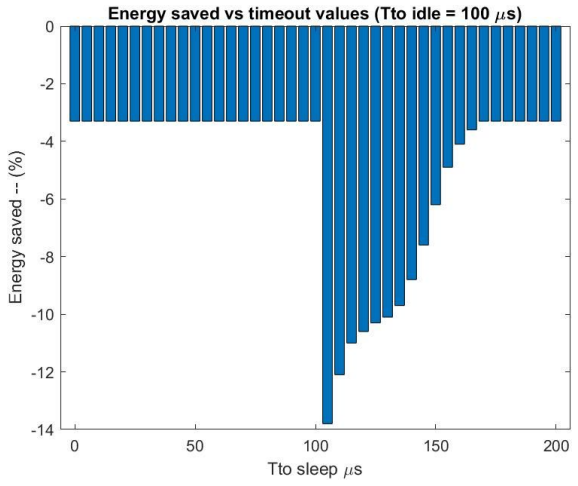


Average time in states:

Run: 1.432038s
Idle: 0.049093s
Sleep: 0.013835s

Figure 23

Workload Profile 5 - Idle times from a trimodal distribution

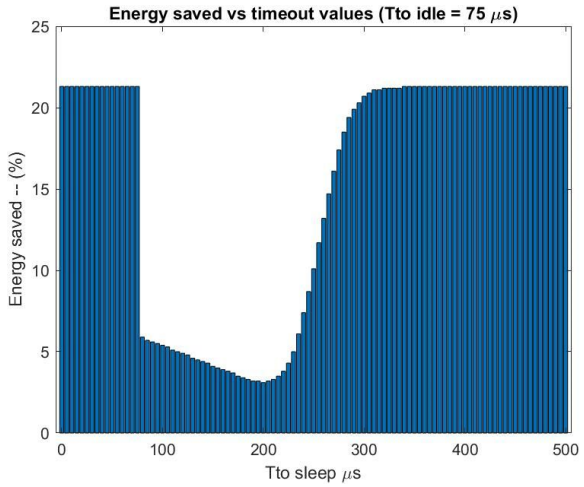


Average time in states:

Run: 1.581207s
Idle: 0.137829s
Sleep: 0.035216s

Figure 24

Custom Profile 1

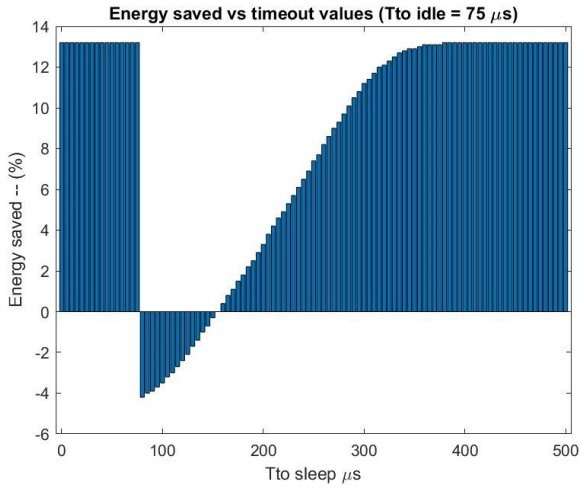


Average time in states:

Run: 2.191702s
Idle: 0.284264s
Sleep: 0.060078s

Figure 25

Custom Profile 2



Average time in states:

Run: 2.017007s
Idle: 0.195594s
Sleep: 0.038832s

Figure 26

Observations

The implemented policies show good results only for the custom workloads. On the other hand, an increment in power consumption is shared by all the other workload types, excluding the uniform low utilization one. Overall we have positive impacts on power savings thanks to the additional *sleep* low power state.

History based policy

Introduction

The main disadvantage of using a timeout policy is the power wasted while waiting for the timeout to expire. A possible solution to reduce the waste is to use a history based policy in which a prediction of the incoming idle periods is computed based on past time values. If the system finds that the predicted idleness will be long enough, it will move to the low power state. The prediction, whose accuracy is crucial for good energy savings, is made according to the past *Tidle* and *Tactive* times. In order to emulate a

correlation between consecutive time samples, additional correlated workload profiles have been generated using the *generate_correlated_workload.m* MATLAB script. Different models, with different polynomials exist to compute the prediction. In this case, a second grade polynomial is chosen. The expression to evaluate the time idleness is shown below.

$$T_{pred}(x) = K_1 T_{idle}[i-1]^2 + K_2 T_{idle}[i-2] + K_3$$

The coefficients are computed for each correlated workload type by applying fitting function using MATLAB scripts. Then, excluding the exponential workload, which parameters are very different from the others, they are meant together, providing the result below.

$K_1 = -6.82851e10$ $K_2 = 1.0003$ $K_3 = 2.13$

The same procedure is applied to obtain the correlation parameters for the custom workloads.

Custom workload 1:

$K_1 = 7.3631e-5$ $K_2 = -0.03631$ $K_3 = 260.9$

Custom workload 2:

$K_1 = 3.0271e-5$ $K_2 = 0.3221$ $K_3 = 131.8$

In order to simulate the history based policy, the *dmp_simulator* has been further modified. The idleness predicted time is computed using the above formula. First, the simulator compares the *idle* state time with the prediction, if it is greater, the system will move to the *idle* state, otherwise it will remain in the *run* state. The same happens with the *sleep* state threshold.

The correlated workload profiles

Uniform distribution Each sample is generated from a uniform distribution whose parameters are obtained as the sum or difference between the previously extracted sample and the middle value of the previous distribution.

Normal distribution | Exponential distribution | Tri-Modal distribution Each extracted sample is used as the mean value for the next distribution. The standard deviations are kept the same. In case of a negative sample, the extraction is repeated until a positive value is generated.

Scripts for automatic power analysis

To automatise the simulation process, multiple bash scripts have been used:

scriptLaunchHistoryPredictionCustomWls.sh *scriptHistoryPredictionCustomWls.sh*
scriptLaunchHistoryPredictionCorrelated.sh *scriptHistoryPredictionCorrelated.sh*
scriptLaunchHistoryPredictionUncorrelated.sh *scriptHistoryPredictionUncorrelated.sh*

The *launch* version script executes the *main* script with different *Tto_idle* values according to the specific workload profiles. The *main* script launches multiple simulations with different sleep timeout threshold, sweeped by 5 µs each time and the

history policy parameters coming from the fitting analysis of the workloads. At the end, it gathers all the power consumption statistics. As before, MATLAB scripts are used to generate graphical representations of power saved with respect to timeout values and the average time spent in each state, for each workload. The average timings are presented for the custom workload only. The same approach is applied to the correlated, uncorrelated and custom workloads. In the following, the graphical representations of the correlated workload are provided.

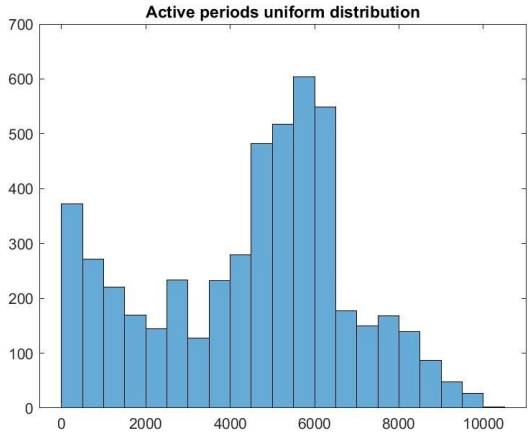


Figure 27

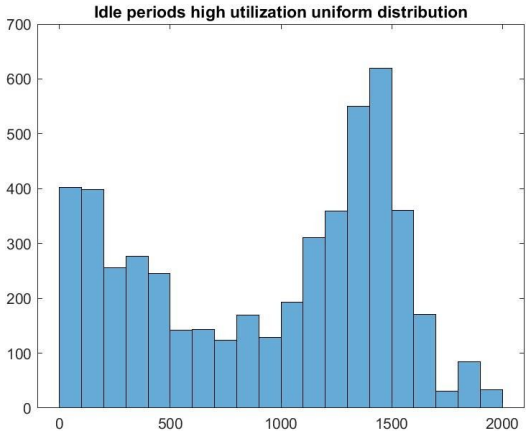


Figure 28

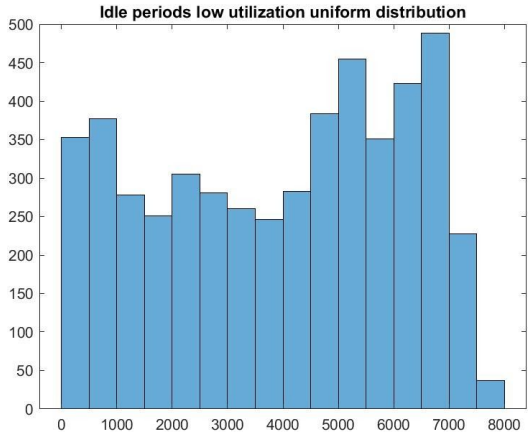


Figure 29

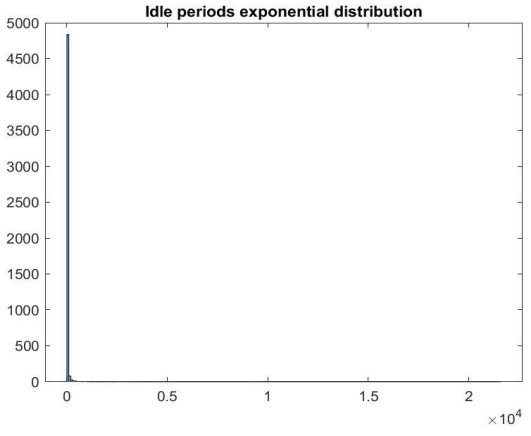


Figure 30

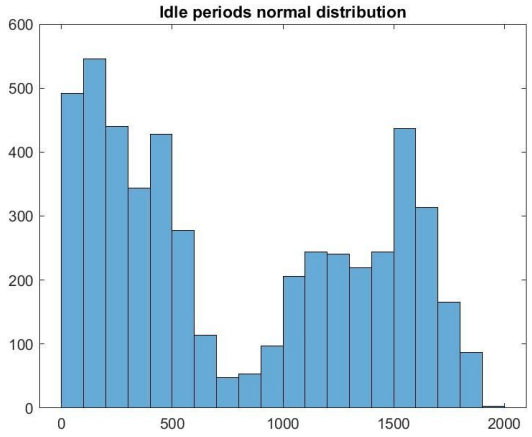


Figure 31

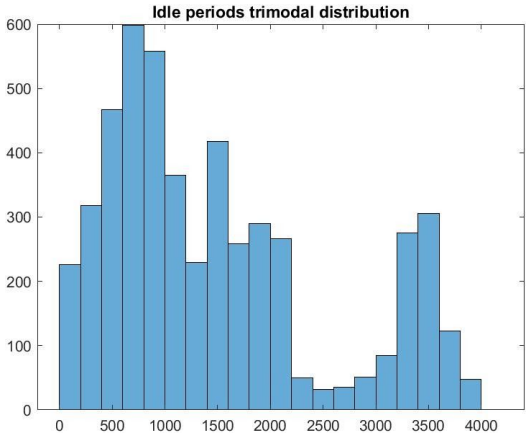


Figure 32

Experimental results

As happened for the timeout policies applied to the full PSM version, given the high number of simulations, a subset of all the possible simulations is provided below. In addition, to underline the differences when a predictive policy is applied on uncorrelated workloads, for each simulation below, the correspondent power savings for the same type of uncorrelated workloads is given in the right side.

Workload Profile 1 - Idle times from a uniform high utilization distribution

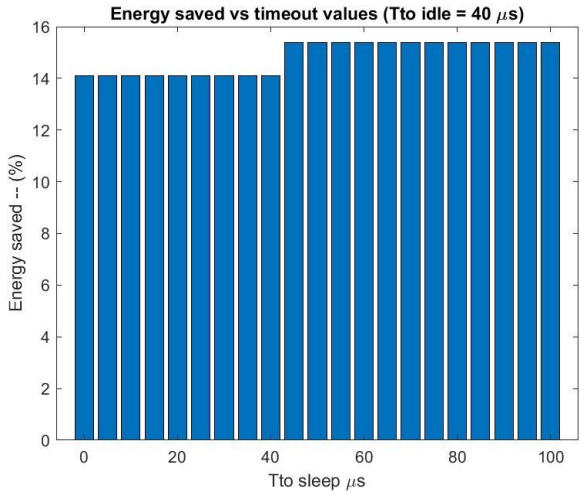


Figure 33

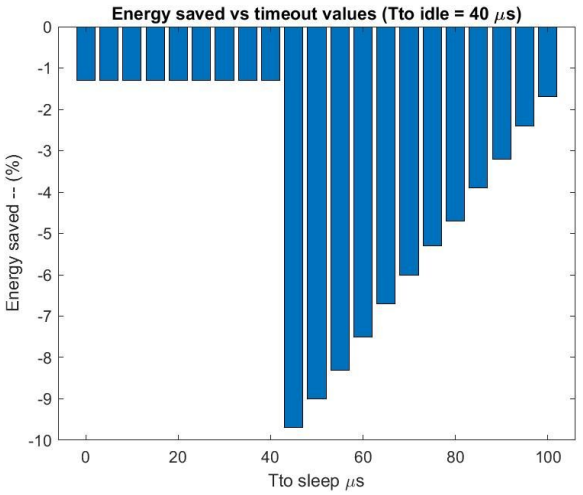


Figure 34

Workload Profile 2 - Idle times from a uniform low utilization distribution

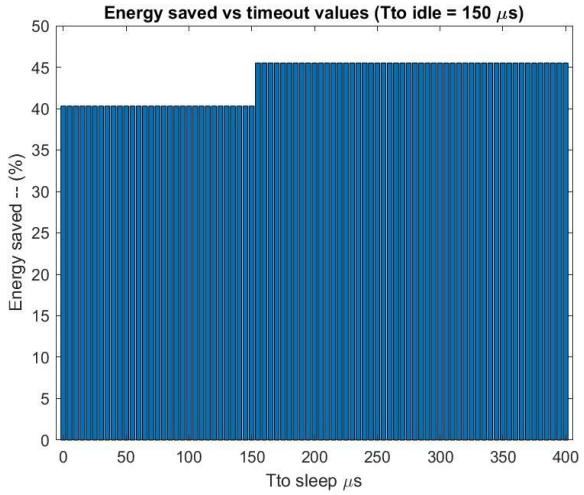


Figure 35

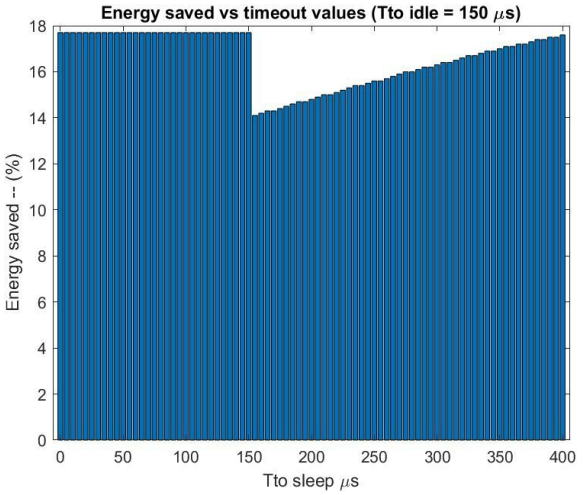


Figure 36

Workload Profile 3 - Idle times from a normal distribution

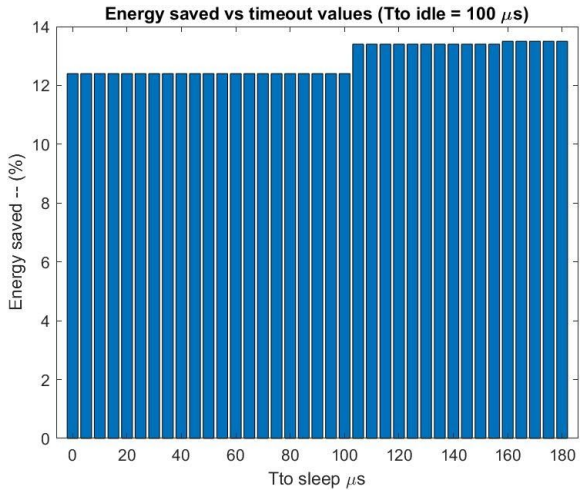


Figure 37

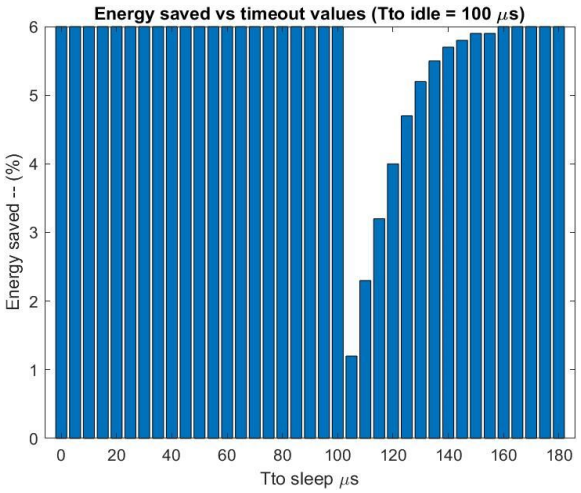


Figure 38

Workload Profile 4 - Idle times from an exponential distribution

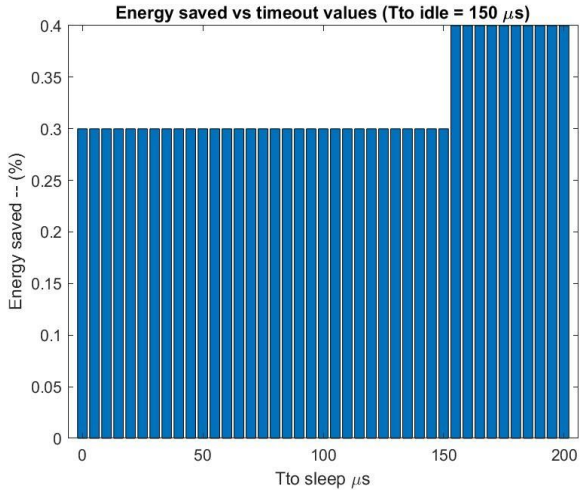


Figure 39

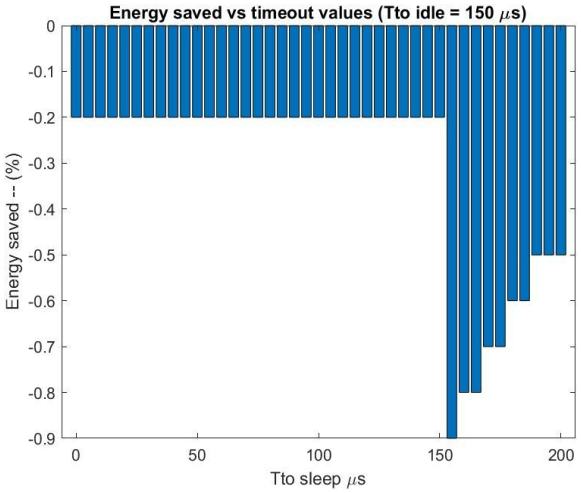


Figure 40

Workload Profile 5 - Idle times from a trimodal distribution

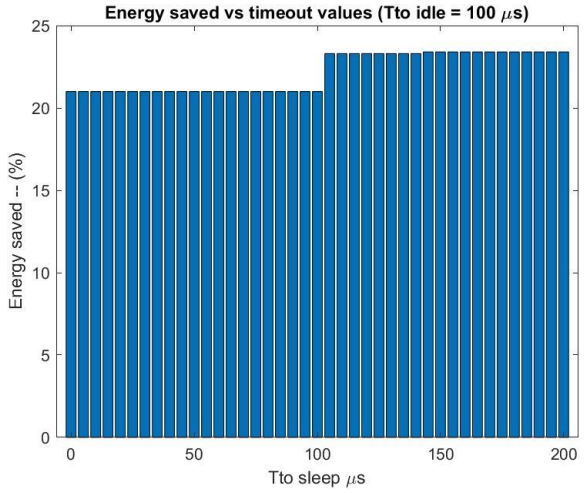


Figure 41

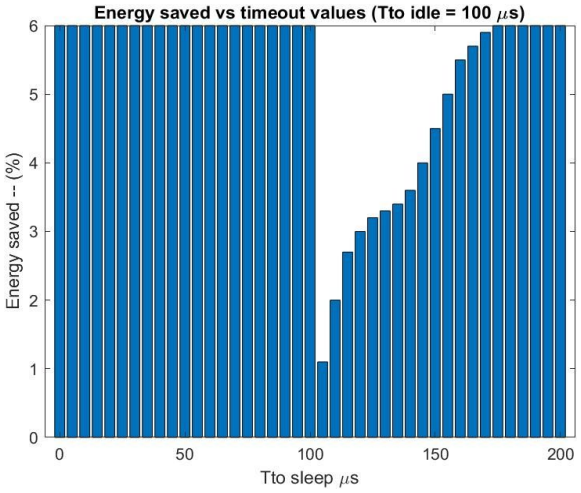


Figure 42

Custom Profile 1

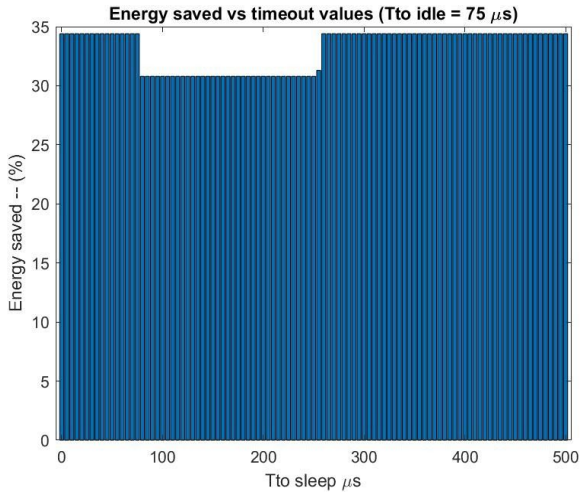


Figure 43

Custom Profile 2

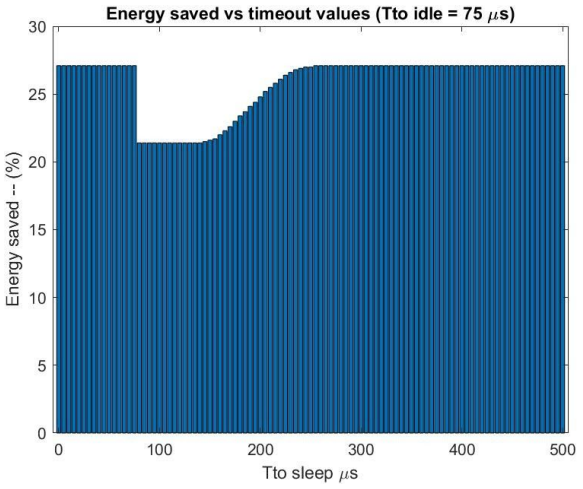


Figure 44

Workload profile	Run (s)	Idle (s)	Sleep (s)
Custom 1	1.864390	0.499355	0.172299
Custom 2	1.853735	0.314909	0.082787

Observations

The use of a predictive policy shows positive results for every condition and profile. A common regularity of the results is shared by all the above graphs. Even the simulations for the uncorrelated workloads that have been performed using the same prediction parameters of the correlated versions, show, in the majority cases, some energy savings. Similar considerations can be performed for the custom workloads. A higher amount of energy savings for the majority of timeout parameters is reported with respect to the timeout policy. The only case that provides poorer results with respect to the average is the exponential workload. The reason is likely related to the fact that the prediction parameters have been generated without considering the exponential distribution. If they were considered, we would surely have less overall energy savings for the other profiles. In addition, a further element has to be considered. Even if the predictive policy shows good energy savings, it requires additional processing and comparisons to be realized. For this reason, a trade off between the complexity of the prediction and the quality of the results should be carefully evaluated.

Other experiments on timeout policy

The timeout based policy is further tested with the correlated workloads used to test the history policy. The graphical results regarding the same *Tto_idle* values presented before, are presented in the following.

Workload Profile 1

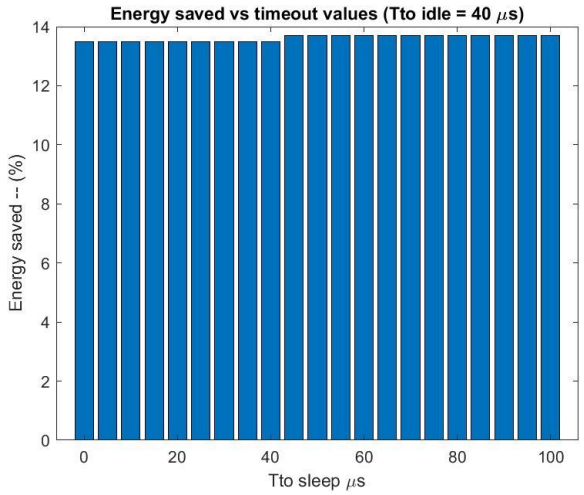


Figure 45

Workload Profile 2

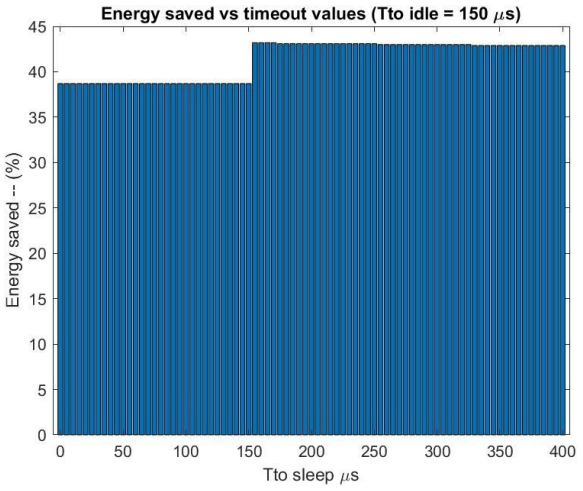


Figure 46

Workload Profile 3

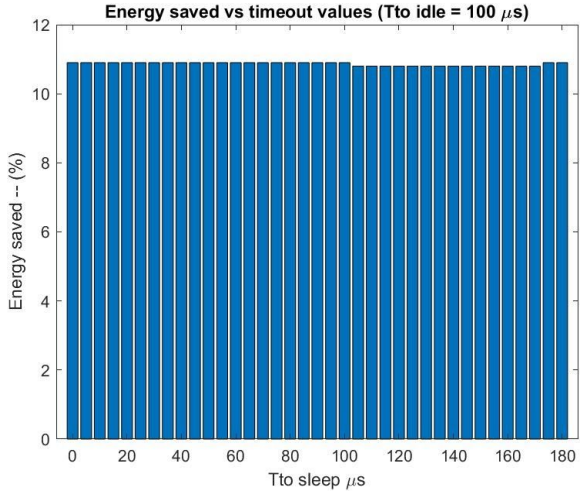


Figure 47

Workload Profile 4

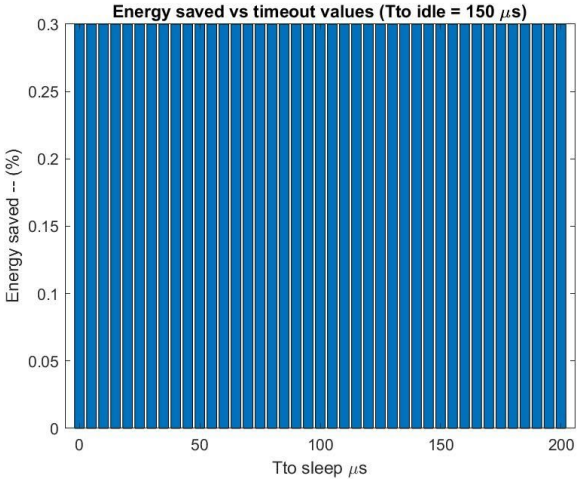


Figure 48

Workload Profile 5

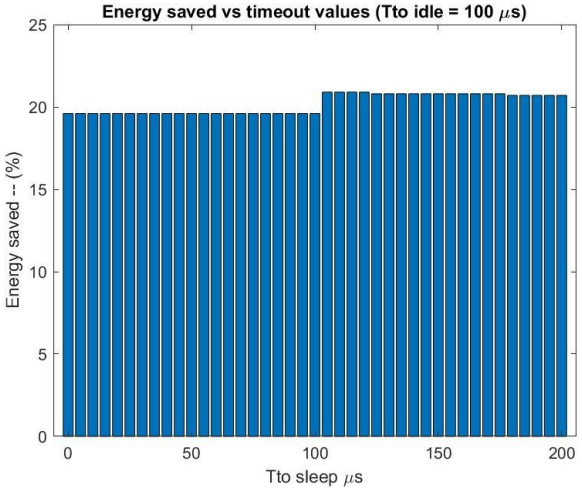


Figure 49

Observations

It appears that the timeout policy gives better results with correlated workloads rather than with the uncorrelated ones. We can notice both higher and more uniform percentages of energy savings. As an example, *figure 20* and *figure 45*, shows the results referred to the timeout policy with uncorrelated and correlated workloads for the distribution profile 1. The comparison shows that, in the latter case, the energy savings are higher. Similar results are obtained for the other workloads. Overall, the timeout policy always shows positive energy savings, meaning that DPM does not anymore make the system waste more power as no DPM was used.

Other experiments on history policy

Further experiments with different types of regression models have been conducted to test the goodness of the history policy. Coefficients from first (referred as *Fit V1*) and a third (referred as *Fit V2*) polynomial grade fittings are generated. The predicted illnesses are generated according to such polynomial models. The simulations, applied to the correlated, the uncorrelated and the custom workloads, are performed following the same structure previously presented. In the following, the obtained coefficients along with the idleness prediction formulas, are provided. In addition, for the custom workloads only, the average time spent in each state is presented.

First grade prediction idleness:

$$T_{pred}(x) = K_1 T_{idle}[i-1] + K_2$$

Average of the coefficients (excluding workload 4):

$K_1 = 0.9990 \qquad K_2 = 2.6799$

Custom Workload 1:

$K_1 = 0.001497 \qquad K_2 = 256$

Custom Workload 2:

$K_1 = 0.3339 \qquad K_2 = 130.8$

Third grade prediction idleness:

$$T_{pred}(x) = K_1 T_{idle}[i-1]^3 + K_2 T_{idle}[i-2]^2 + K_1 T_{idle}[i-3] + K_4$$

Average of the coefficients (excluding workload 4):

$K_1 = -8.7e-9 \qquad K_2 = 1.0736e-4 \qquad K_3 = 0.7631 \qquad K_4 = 9.4413$

Custom workload 1:

$K_1 = 1.559e-5 \qquad K_2 = -0.01199$
$K_3 = 3.045 \qquad K_4 = 0.8927$

Custom workload 2:

$K_1 = -2.868e-7 \qquad K_2 = 0.0001985$
$K_3 = 0.2929 \qquad K_4 = 133.27$

	First grade average times (Fit V1)			Third grade average times (Fit V2)		
Workload type	**Run (s)**	**Idle (s)**	**Sleep (s)**	**Run (s)**	**Idle (s)**	**Sleep (s)**
Custom 1	1.864390	0.49873	0.172901	1.875901	0.602139	0.058004
Custom 2	1.854302	0.314534	0.082596	1.859161	0.345502	0.046769

Observations

Despite the use of a different grade regression model, the average energy savings are very similar to the ones obtained before with the second grade regression. The comparison of the data coming from the average times spent in each state shows that, for *custom 1* workload, the first grade regression makes the system spend a higher time in the *sleep* state than the third grade model. Moreover, the first grade average times are very similar to the ones obtained for the second grade model.

Other experiments

An additional custom workload, referred to as *my_wl,* is considered. Its active periods are the same as the correlated workloads, while the idle ones are generated as a random combination of all the workloads presented at the beginning. The distribution of the active and idle periods are provided in *figure 50* and *figure 51*.

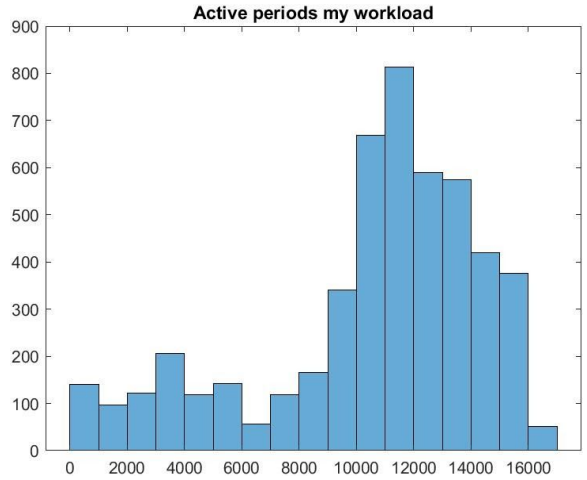


Figure 50

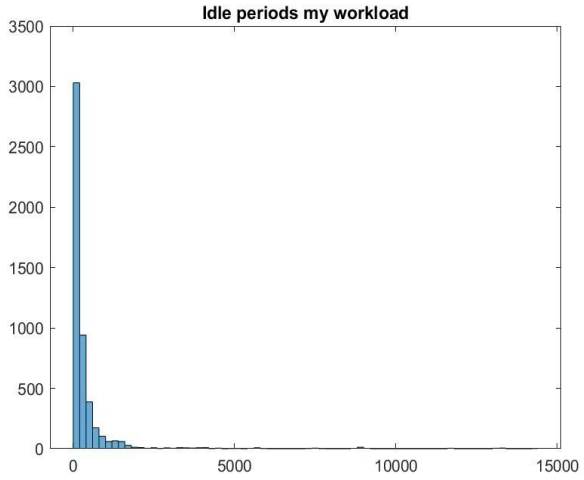


Figure 51

The idle values are very similar to an exponential distribution. The same execution flow is repeated for *my_wl*, except for the additional history policy using *Fit V1* and *Fit V2*, that have demonstrated to not improve the results obtained with the two grade prediction.

Observations

As happened in the majority of previous cases, the use of a DPM has a positive impact on energy savings. On average, the history policy shows better results than the ones obtained with a timeout policy. As an example, in *figure 52* and *figure 53*, the energy savings obtained in case of a timeout and a history policy, when the *idle* timeout is set to

50 µs, are provided. Even if the average amount of energy cost is quite limited, both DPM shows a quite uniform savings for all the *sleep* timeouts. In particular, again, the use of a history based policy, thanks to a higher quality idleness prediction, provides better results in comparison with the ones obtained with a timeout policy.

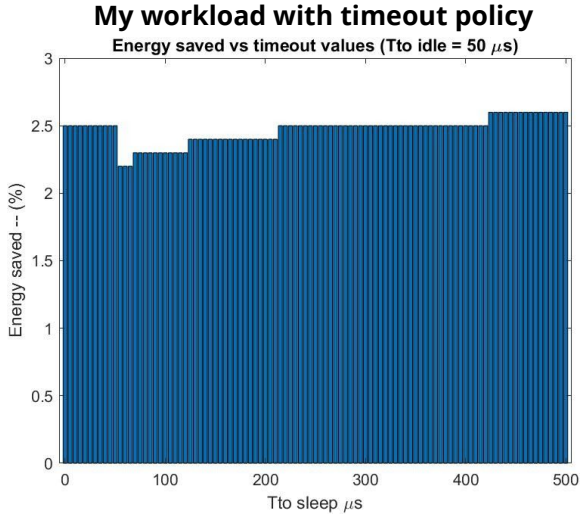


Figure 52

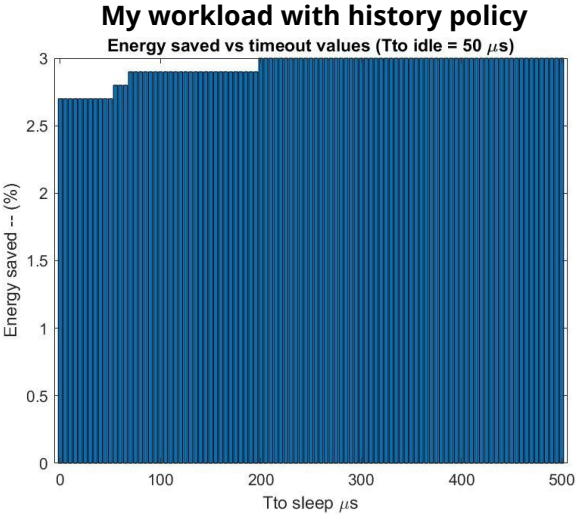


Figure 53

Conclusion

The use of a DPM becomes essential whenever non-ideal transitions are considered. Moving among different states has a cost in both power and time. The central DPM goal is the understanding, according to static or dynamic parameters, how long the next idleness will be. Based on this prediction, the system knows whether it is convenient to move into a low power state. In this experience, we have demonstrated that the adopted distribution types have different impacts on the quality of the DPM. Moreover, different prediction methods affect the quality of the DPM, hence the amount of saved energy. Two static DPM policies have been implemented, a timeout based and a history based. The first mainly provides positive results for low utilization profiles. In the case of a single low power state, it happens that it shows worse energy consumption with respect to a solution without dynamic power management. On the other hand, good results in all conditions are obtained by using the history based DPM. Moreover, for the presented workloads, different regression models to feed the history based DPM, do not seem to improve energy savings but only slightly modify the average time spent in each state. Despite that, for all the workloads, it has been shown that the use of a history policy offers higher energy savings with respect to a timeout DPM. However, the good quality of such policy is directly associated with knowing a priori the shape of the workloads. For this reason, in order to get positive results in more realistic cases, for which the workloads are usually unknown or nonstationary, dynamic policies like adaptive or stochastic should be taken into account. For those advanced solutions, the additional overhead due to their complexity will have a non-negligible impact on the additional required computations but the result will be closer to an optimal one.