


Dynamic Power Management: an Introduction

Alessandro Bogliolo

Information Science and Technology Institute
University of Urbino
61029 Urbino, Italy
bogliolo@sti.uniurb.it

Dynamic Power Management (DPM)

1. Computer systems are designed to deliver peak performance, but are often idle or used to perform tasks that do not require such performance
2. For a given technology/architecture, there is a close relation between the performance provided by a system and the power it consumes
3. Dynamic power management (DPM) is a control strategy aimed at adapting the power/performance of a system to its workload


Dynamic Power Management: an Introduction
scope

Information Science
Technology Institute

DPM at FSM-05-Moby (0)


Formal Methods for Mobile Computing

- Why do we need to apply *formal methods* to DPM?
- Why do we need to apply DPM to *mobile computing*?

© alessandro bogliolo

sfm-05 moby

3/42


Dynamic Power Management: an Introduction
scope

Information Science
Technology Institute

DPM at FSM-05-Moby (1)


Formal Methods for Mobile Computing


- *Formal methods* for the design of computer, communication and software systems
- Power consumption is a key design metric for today's information systems
 - it affects cost, performance, size, reliability, usability, ...
- Low-power design techniques are often applied to the development of HW/SW components
- Big power savings can be achieved at run time by means of DPM

© alessandro bogliolo

sfm-05 moby

4/42



Dynamic Power Management: an Introduction
scope





DPM at FSM-05-Moby (2)

*Formal Methods for **Mobile Computing***

- **Mobile systems** are battery-powered
- Power consumption directly affects availability
- **Mobile systems** use wireless channels to communicate
- The wireless network interface is responsible for a large fraction of the power consumption of portable devices (cellular phones, PDAs, laptops, ...)
- Wireless protocols support DPM


© alessandro bogliolo
sfm-05 moby
5/42


Dynamic Power Management: an Introduction
scope




DPM at FSM-05-Moby (3)


*Formal Methods for **Mobile Computing***


- Formal methods are mainly focused on functionality and performance
- Power consumption cannot be neglected when designing a mobile system
- DPM may affect both functionality and performance

“Formal Methods for Predicting the Impact of Dynamic Power Management”
A. Acquaviva, A. Aldini, M. Bernardo, A. Bogliolo, E. Bontà, E. Lattanzi

“Dynamic Power Management Strategies Within the IEEE 802.11 Standard”
A. Acquaviva, E. Bontà, E. Lattanzi



© alessandro bogliolo
sfm-05 moby
6/42



Dynamic Power Management: an Introduction
outline




Outline

- Dynamic power management
 - Motivational example
 - Problem statement
 - System components
- Power state machine (PSM)
 - Definition
 - Examples
- Exploitability of low-power states
 - Inherent exploitability
 - Dealing with uncertainty
- DMP strategies
 - Timeout-based shut down
 - Preemptive wakeup
 - Stochastic control
- Open issues / Advanced solutions



© alessandro bogliolo
sfm-05 moby
7/42

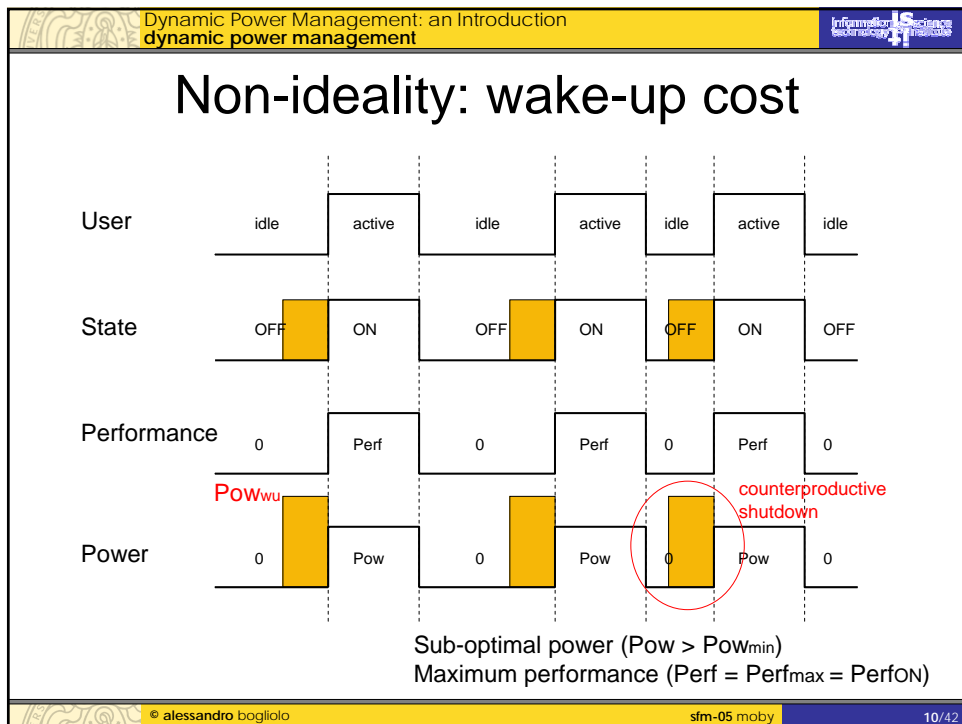
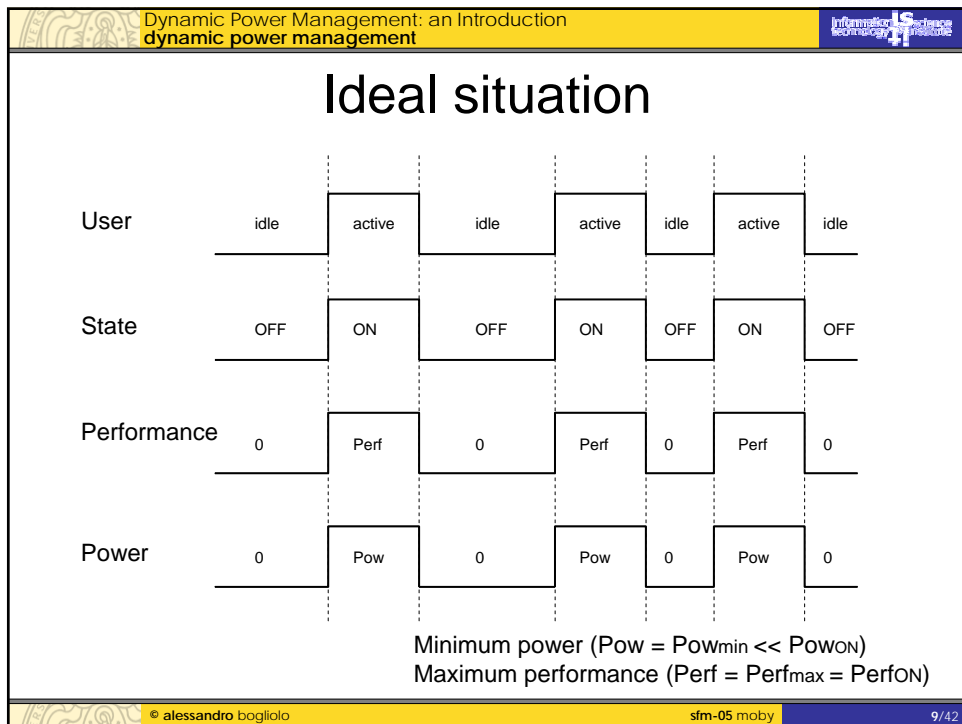

Dynamic Power Management: an Introduction
dynamic power management



Motivational example

- The simplest example of a power-manageable system is a device that can be dynamically switched ON and OFF
- When ON:
 - It provides a given performance (PerfON)
 - It consumes a constant amount of power (PowON)
- When OFF:
 - It provides NO performance
 - It consumes NO power
- To save power, the device should be ON whenever it is used and OFF whenever it is not.


© alessandro bogliolo
sfm-05 moby
8/42



Non-ideality: misprediction

The diagram illustrates the timing of dynamic power management across four dimensions: User, State, Performance, and Power. Vertical dashed lines mark the boundaries of user activity.

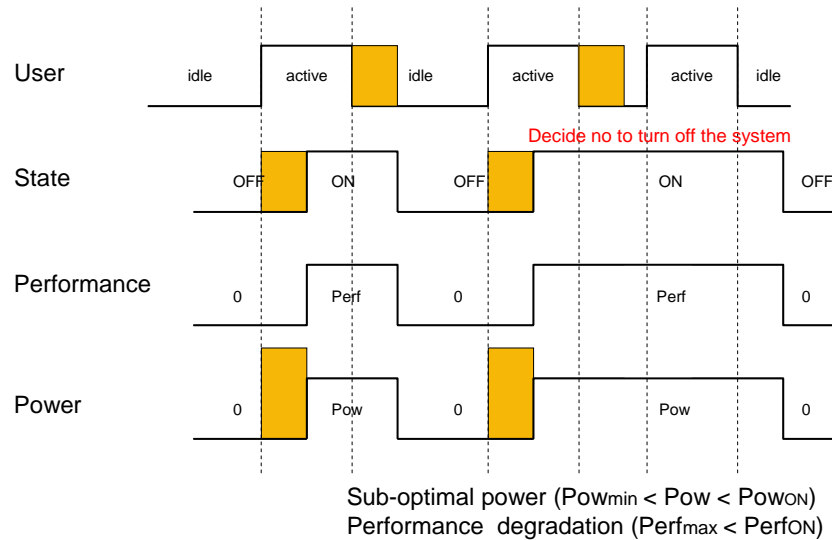
- User:** Shows periods of 'idle' and 'active' usage. Active periods are highlighted with yellow blocks.
- State:** Shows the system state as 'OFF' or 'ON'. Transitions from OFF to ON occur at the start of active periods.
- Performance:** Shows performance levels as '0' or 'Perf'. The transition to 'Perf' occurs after a 'perceived delay' following the state transition.
- Power:** Shows power levels as '0' or 'Pow'. The transition to 'Pow' occurs after the state transition. A 'counterproductive shutdown' is indicated by a red circle around a period where the power is '0' while the user is active.

Sub-optimal power ($Pow > Pow_{min}$)
Performance degradation ($Perf < Perf_{max}$)

Problem statement

- In real systems:
 - State transitions have a non-negligible cost (in terms of time and power)
 - The workload is unknown
- As a consequence:
 - A greedy DPM strategy may be counter-productive
 - DPM may cause a performance degradation
- DPM goal:
 - Dynamically adapt the operating mode of a system to its workload in order to minimize the average power consumption under given performance constraints

DPM example



System components

- Power-manageable resources (server, S)
 - Provides a given functionality (service)
 - Provides different power-performance tradeoffs
 - May dynamically change the power-performance tradeoff upon external DPM commands
- Workload (user/client, C)
 - Issues service requests to the server
- Power manager (PM)
 - Observes the state of the system and the workload
 - Controls the power-performance tradeoff of the server by issuing DPM commands according to a given *DPM policy*

Outline

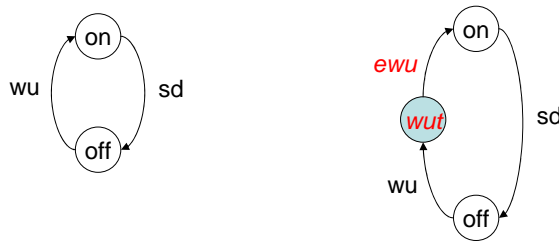
- Dynamic power management
 - Idleness
 - Problem statement
 - System components
- Power state machine (PSM)
 - Definition
 - Examples
- Exploitability of low-power states
 - Inherent exploitability
 - Dealing with uncertainty
- DMP strategies
 - Timeout-based shut down
 - Preemptive wakeup
 - Stochastic control
- Open issues / Advanced solutions

Power state machine (PSM)

- A PSM is a (F)SM whose states (called *power states*) represent the operating modes of a system, characterized only by:
 - Average performance (*Perf*)
 - Average power consumption (*Pow*)
- *Active* states
 - Provide non-null performance
- *Inactive* states
 - Provide no performance
- Transitions among states
 - Are triggered either by internal or by external events
 - Are characterized by *transition time* and *energy*

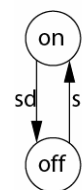
State diagrams

- PSMs can be represented as state diagrams
- If transition times are not implicitly represented as properties of the edges of the state diagram, they need to be explicitly represented by transient states with outgoing transitions triggered by self-events



Example (1)

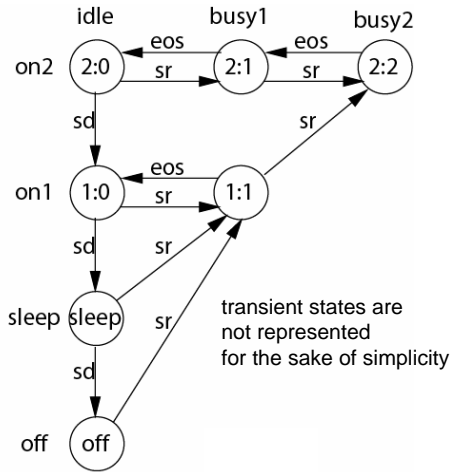
- A server with only 2 power states:
 - On (active), with $(Perf, Pow) = (Perf_{on}, Pow_{on})$
 - Off (inactive), with $(Perf, Pow) = (0, 0)$
- A client issuing service requests (sr)
- A DPM issuing shut-down commands (sd)
- Shutdown transitions are triggered by sd commands
- Wakeup transitions are triggered by service requests



transient states are not represented for the sake of simplicity

Example (2)

- A server with 2 equivalent processing units that can be progressively switched ON and OFF:
- The PSM has:
 - 2 active states
 - 2 inactive states
- The state diagram also shows idle/busy condition to model the service time and to make the server sensitive to sd commands only when idle



Outline

- Dynamic power management
 - Idleness
 - Problem statement
 - System components
- Power state machine (PSM)
 - Definition
 - Examples
- Exploitability of low-power states
 - Inherent exploitability
 - Dealing with uncertainty
- DMP strategies
 - Timeout-based shut down
 - Preemptive wakeup
 - Stochastic control
- Open issues / Advanced solutions

Inherent exploitability

- Any component needs to be active to provide a service
- The *transition cost* of an inactive state is the sum of the costs of the shutdown and wakeup transitions from and to the closest active state

$$\begin{aligned} T_{TR} &= T_{SD} + T_{WU} \\ E_{TR} &= E_{SD} + E_{WU} \end{aligned} \quad P_{TR} = E_{TR} / T_{TR}$$

- Transition cost limits the exploitability of inactive states
- If $P_{TR} > P_{ON}$ entering the inactive state is convenient iff the idle time is long enough to compensate for the transition cost

Breakeven time

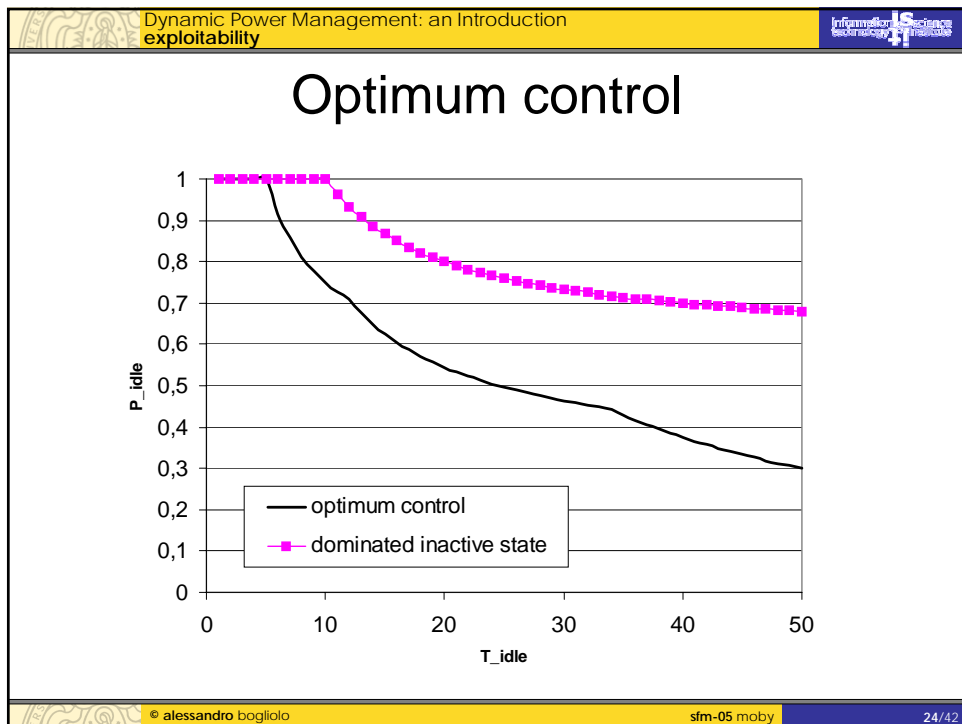
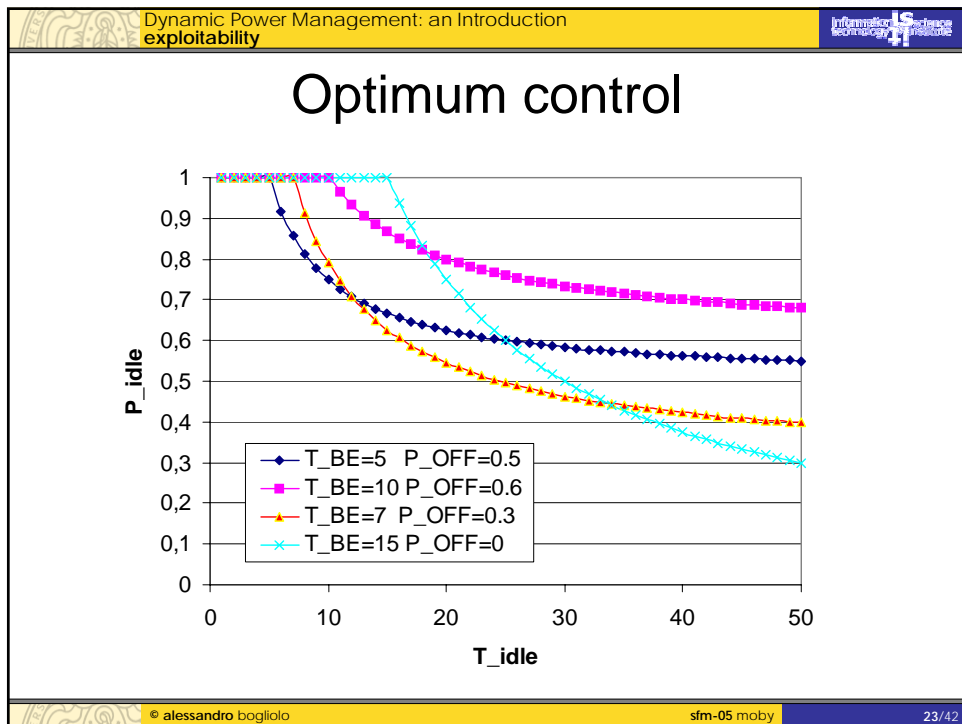
Minimum idle period that makes it convenient to go to an inactive state

- If $P_{TR} > P_{ON} > P_{OFF}$
- $$(T_{BE} - T_{TR})(P_{ON} - P_{OFF}) = T_{TR}(P_{TR} - P_{ON})$$
- saved energy extra transition energy

$$T_{BE} = T_{TR} + T_{TR} \frac{P_{TR} - P_{ON}}{P_{ON} - P_{OFF}}$$

- An inactive state is exploitable when the idle period is longer than T_{BE}

$$P_{saved}(OFF, T_{idle}) = (P_{ON} - P_{OFF}) \frac{T_{idle} - T_{BE}}{T_{idle}}$$



Best oracle

- The best-oracle is an ideal DPM that takes optimal decisions based on a complete (*a priori*) knowledge of the workload
- In case of a system with a unique inactive state (OFF), the best oracle achieves the following savings at no cost:

$$P_{\text{saved,OFF}} = \frac{\int_{T_{\text{idle}}=T_{\text{BE}}}^{\infty} (P_{\text{ON}} - P_{\text{OFF}})(T_{\text{idle}} - T_{\text{BE}}) \text{prob}(T_{\text{idle}}) dT_{\text{idle}}}{T_{\text{idle}}^{\text{avg}}}$$

$$P_{\text{saved,OFF}} = (P_{\text{ON}} - P_{\text{OFF}}) \frac{T_{\text{idle} > T_{\text{BE}}}^{\text{avg}} - T_{\text{BE}}}{T_{\text{idle}}^{\text{avg}}} (1 - F_{T_{\text{idle}}}(T_{\text{BE}}))$$

Dealing with uncertainty

- The best oracle is an ideal DPM
- In real cases the DPM has no a-priori knowledge of the length of incoming idle periods
- As a consequence:
 - It cannot guarantee to take always optimal decisions
 - It cannot guarantee to wakeup the system in time to serve upcoming requests with no delay
- Workload prediction is one of the most challenging issues of DPM

Workload prediction

- Problem statement:
predict an incoming event (e) based on the observation of a past event (o) under the assumption that
 $Prob(e|o) > Prob(e)$

- A *safe* predictor never makes over-predictions:

$$Prob(e|o)=1$$

- An *efficient* predictor never makes under-predictions:

$$Prob(o|e)=1$$

Outline

- Dynamic power management
 - Idleness
 - Problem statement
 - System components
- Power state machine (PSM)
 - Definition
 - Examples
- Exploitability of low-power states
 - Inherent exploitability
 - Dealing with uncertainty
- DMP strategies
 - Timeout-based shut down
 - Preemptive wakeup
 - Stochastic control
- Open issues / Advanced solutions

Classification criteria

- DPM strategies can be classified based on:
 1. The *predictor* they use
 2. The *degree of control* granted to the power manager
 3. The *nature of the decision* taken by the power manager

Classification criteria: Predictors

- Target
 - Occurrence probability of an idle period longer than T_{BE}
 - Length of next/current idle period T_{idle}
- Observed history
 - Average length of the last n idle periods
 - Length of the last activity burst
 - First part of the target idle period

Classification criteria: Degree of DPM control

- Reactive DPM
 - Shuts down the system when user's idleness is detected
 - Has no control on wake up
- Proactive DPM
 - Usually shuts down the system when user's idleness is detected
 - Has control on wake up
 - Attempts to preemptively wake up the system to serve next service request with no delay

Classification criteria: Nature of DPM decisions

- Deterministic DPM policies
 - Take deterministic decisions based on the observed (current) state of the workload and of the system
 - The same decision is taken whenever the same conditions occur
- Stochastic/Randomized DPM policies
 - Take randomized decisions whose probability distribution is based on the current state
 - Different decisions may be taken corresponding to the same conditions

Timeout-based shut-down (1)

- Predictor:
 - Observe the first part of the current idle period to predict the length of the remaining part
$$e = \{T_{idle} > T_{BE} + T_{to}\} \quad o = \{T_{idle} > T_{to}\}$$
- Degree of control:
 - Reactive (issues only shut-down commands)
- Nature:
 - Deterministic DPM

Timeout-based shut-down (2)

- Applicability:
 - Good correlation between observed (o) and predicted (e) events
 - Depends on the first-order distribution of idle periods
 - Exponential distributions are memory-less (timeout is ineffective)
 - Bi-modal distributions are a best case
 - Deterministic idle periods are worst case
- Cons:
 - Wasted energy
 - Longer breakeven time

Preemptive wakeup (1)

- Predictor:
 - Observe workload history to predict T_{idle}
- Degree of control:
 - Proactive
 - Issues both shut-down and wake-up commands
 - Attempts to wake up the system in time to serve upcoming requests with no delay (i.e., at $T_{idle} - T_{WU}$)
- Nature:
 - Deterministic DPM

Preemptive wakeup (2)

- Applicability:
 - Good time correlation of the workload
 - Negative correlation between T_{idle} and the length of the last activity burst
 - Positive correlation between T_{idle} and the last observed idle period
- Cons:
 - Mispredictions may cause
 - Sub-optimal decisions
 - Counter-productive shut-downs
 - Performance degradation

Stochastic control (1)

*Randomized policies provide best solutions
to constrained optimization problems*

Stochastic control (2)

- Example:
 - Reactive shut-down
 - 50% of exploitable idle periods
 - Ideal predictor
 - $T_{WU}=1$
- Possible deterministic policies:
 1. Never shuts down the system
no power saving, no performance degradation
 2. Shut down the system at each exploitable idle period
maximum power saving, average service delay 0.5
 3. Always shut down the system
sub-optimal power saving, average service delay 1

Stochastic control (3)

- Example:
 - Reactive shut-down
 - 50% of exploitable idle periods
 - Ideal predictor
 - $T_{WU}=1$
 - Tolerated average service delay 0.1
- Randomized policy:
 - When an exploitable idle period is detected issue a shut down command with probability 0.2
 - 20% of maximum power saving
 - average service delay 0.1

Stochastic control (4)

- Policy optimization problem:
 - Given the power state machine of the system
 - Given the model of the workload
 - Given the performance constraints
 - Find the randomized policy that minimizes power consumption under performance constraints
- Markov decision processes:
 - Constrained policy optimization can be cast to linear programming (LP) and exactly solved in polynomial time
- Pros:
 - Global view
 - Optimality
- Cons:
 - Real-world systems and workload are not Markov processes

Outline

- Dynamic power management
 - Motivational example
 - Problem statement
 - System components
- Power state machine (PSM)
 - Definition
 - Examples
- Exploitability of low-power states
 - Inherent exploitability
 - Dealing with uncertainty
- DMP strategies
 - Timeout-based shut down
 - Preemptive wakeup
 - Stochastic control
- Open issues / Advanced solutions

Open issues / Advanced solutions

- Open issues:
 - Non stationary workloads
 - Large set of power states
 - Continuous state space
 - User/workload models
- Advanced techniques:
 - Adaptive DPM strategies (workload characterization)
 - Pre-computed policies
 - Runtime policy optimization
 - Feedback-loop
 - Simulation-based optimization
 - Workload-driven (or client-driven, or application-driven) DPM