Hierarchical Power Management with Application to Scheduling

Peng Rong and **Massoud Pedram**

Department of Electrical Engineering
University of Southern California

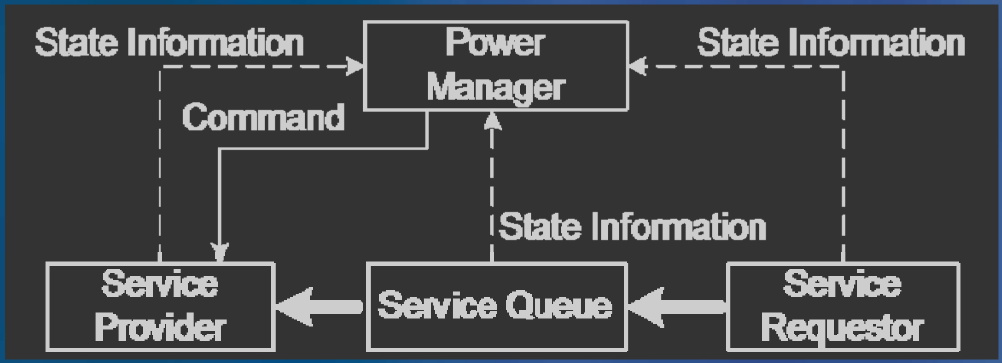
2005 ISLPED

Outline

- Introduction
- Hierarchical Power Management
  - A two-level power management architecture
  - Application scheduling
  - Capturing component dependencies
- System Modeling and Solution Technique
- Experimental Results
- Conclusion

Dynamic Power Management (DPM)

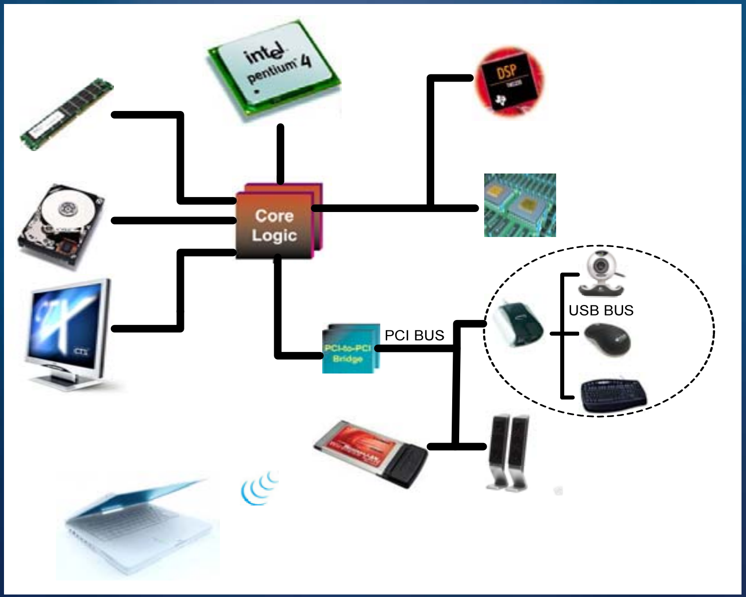
- Principles of operation
  - Selectively shut-down the idle components or slow the underutilized components
  - Adapt system behavior to application needs and available resources



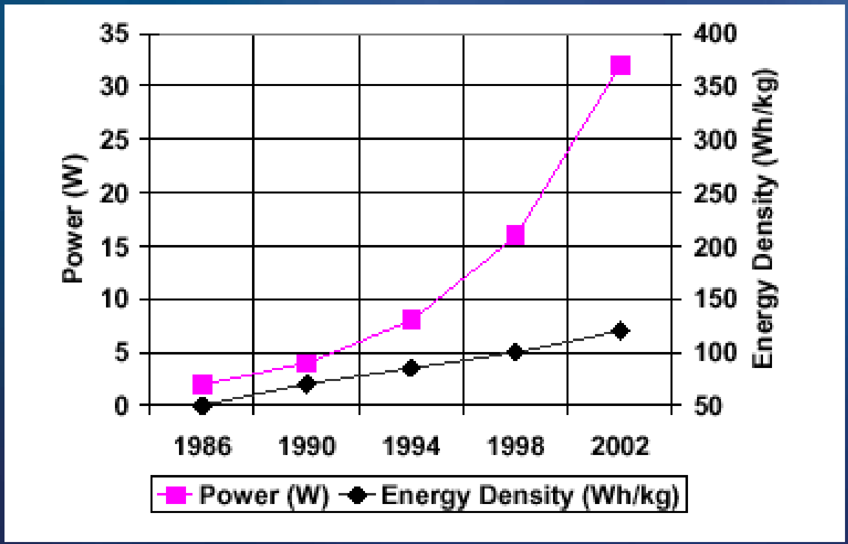
- A power management system
  - Multiple service requestors (SR), multiple service providers (SP), prioritized service queues (SQ)
  - A power manager (PM) monitors the system state and issues commands to shut-down or slow-down SP's
  - Each SP has multiple states which are different in terms of their power dissipation and service speeds

Modern Microelectronic Systems

- Comprise of multiple heterogeneous computing and communication components

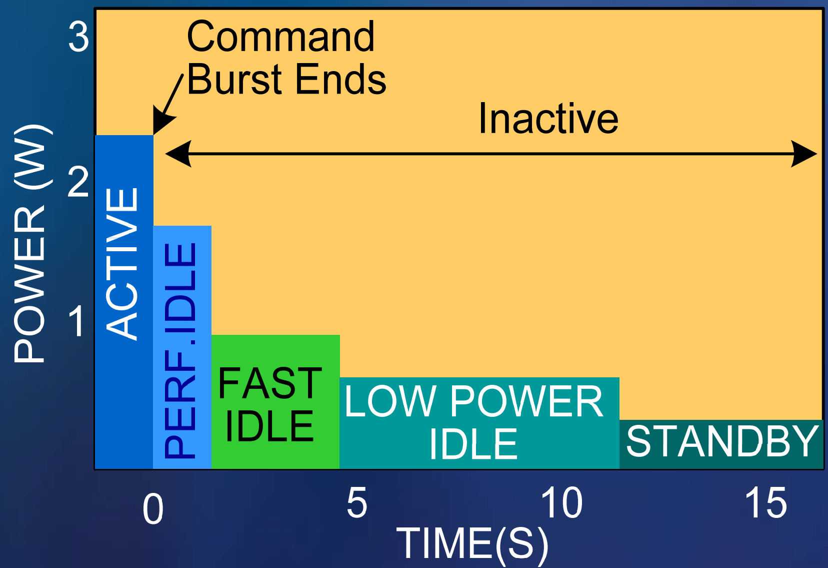


- Power-constrained systems, A widening "battery gap" between system power needs and battery capacity



Self Power-Managed Components

- Protocol-defined power management on data links e.g., USB and power saving mode in IEEE 802.11
- Hardware components with built-in power managers: Enhanced adaptive battery life extender (EABLE) for Hitachi disk drive



A Hierarchal Power Management (HPM) Architecture

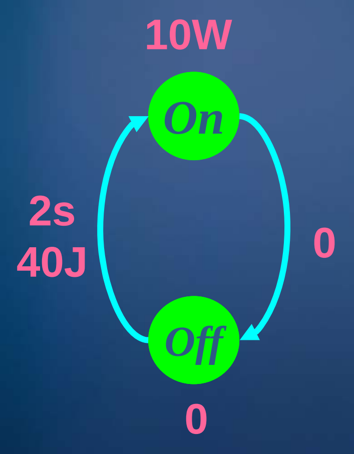
- Supports DPM functions at two levels
  - Component-level functions developed by the IP vendor/manufacturer
    - Simple but generic DPM policy, e.g., timeout policy
  - System-level functions developed by the system designer/integrator
    - Service flow regulation
    - Application scheduling
    - Online tuning of component-level DPM functions
- Benefits
  - Facilitates power-awareness in systems composed of off-the-shelf components and IP blocks
  - Capable of handling hard and/or soft constraints

Prior Work Related to HPM

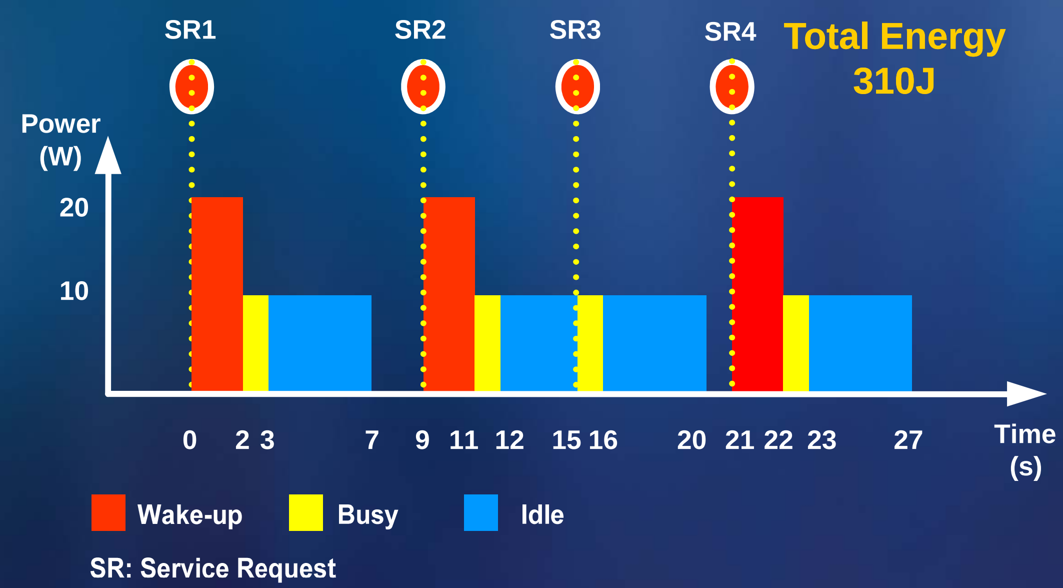
- Managing power consumption in Networks on Chips [Simunic et al. 2002]
  - Network-centric DPM: Source nodes use network sleep/wakeup requests to force sink nodes to enter specified states
- Hierarchical adaptive DPM [Ren and Marculescu, 2003]
  - A hierarchically constructed DPM policy: Seeks an optimal rule to select and employ a policy from among a set of pre-computed ones

Timeout DPM Techniques with an Example

- Timeout policies
  - Start a timer at the beginning of each idle mode; shut down the system if it has been idle for some timeout value
  - Efficient, but can be ineffective because it ignores statistical characteristics of the workload



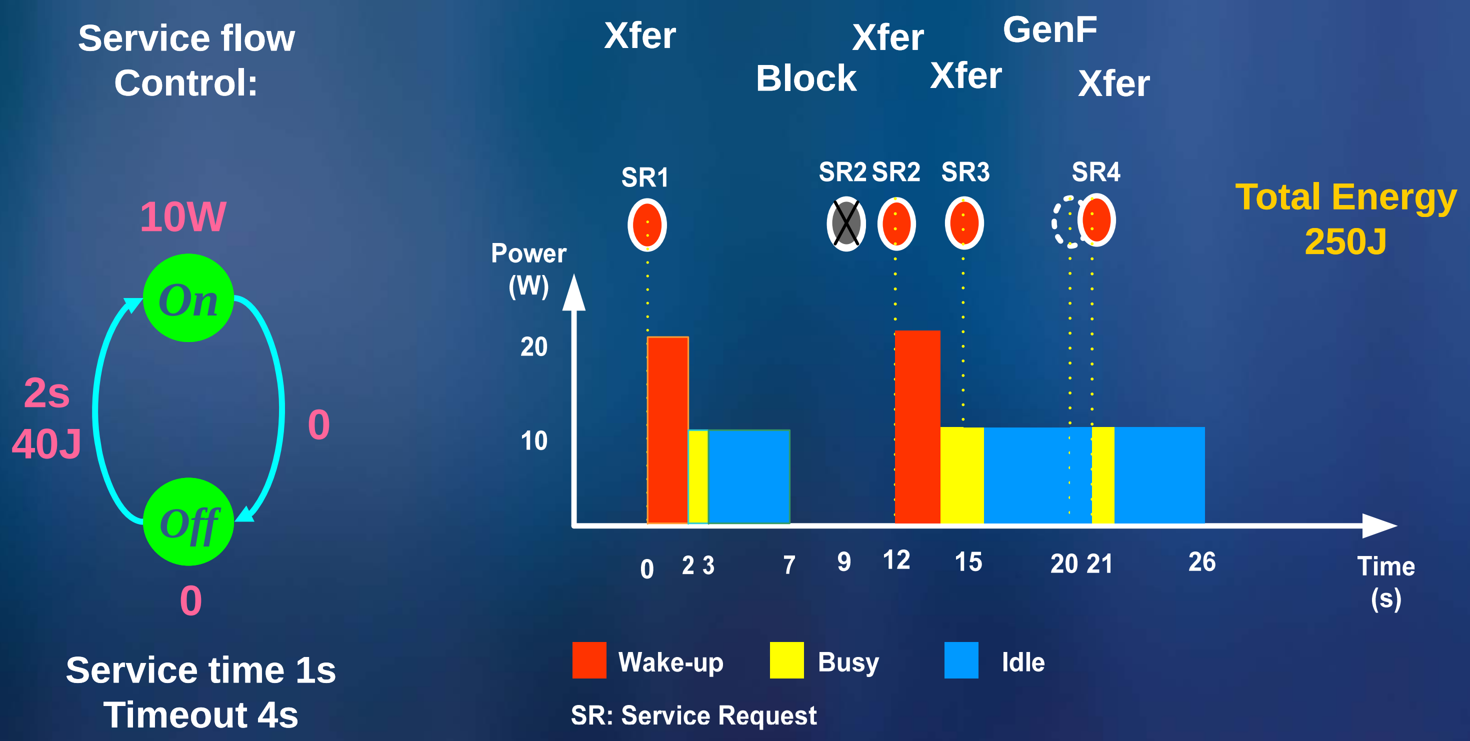
Service time 1s
Timeout 4s



SLPED-05

Service Flow Control for Timeout-based Techniques

- Flow Control Steps
  - Block and Transfer (Xfer)
  - Generate Fake SR (GenF)



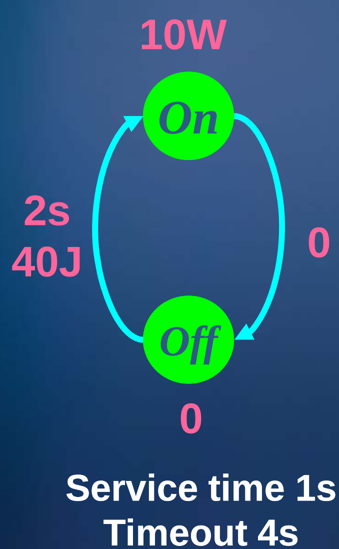
SLPED-05

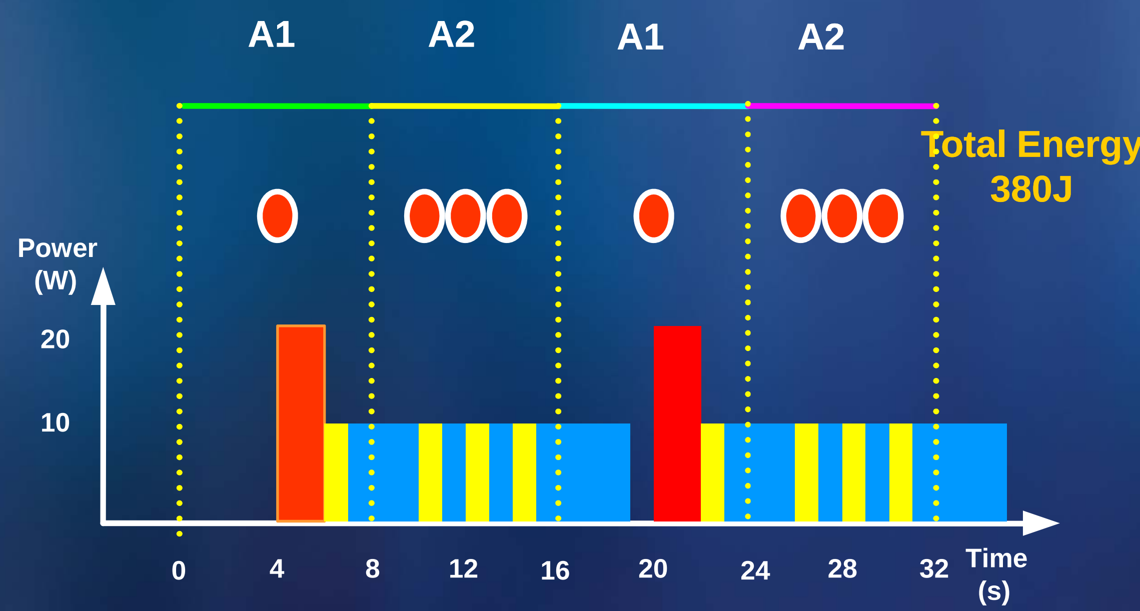
Application Scheduling for DPM: Overview

- Setup: Applications generate SRs to different devices; these SRs tend to have different rates
- Problem statement: Find a job schedule which minimizes the total power dissipation
- Related Work
  - Online job scheduling for low power [Lu et al. 2002]
    - Groups jobs based on their device usage requirements
    - Minimizes power by checking every possible execution sequence of the job groups
- Our Approach
  - Continuous-time Markovian decision process (CTMDP) based application-level scheduling
  - Scheduling is based on states of the individual components, the number of waiting tasks, and characteristics of the application
  - Performed concurrently with DPM optimal policy derivation

Global System State-based Application Scheduling for DPM – An Example

- Two applications A1 and A2 generate SRs
  - A1: 1 SR@8s, A2: 3 SRs@8s
- Without application scheduling
  - Each application is alternately executed for exactly 4s

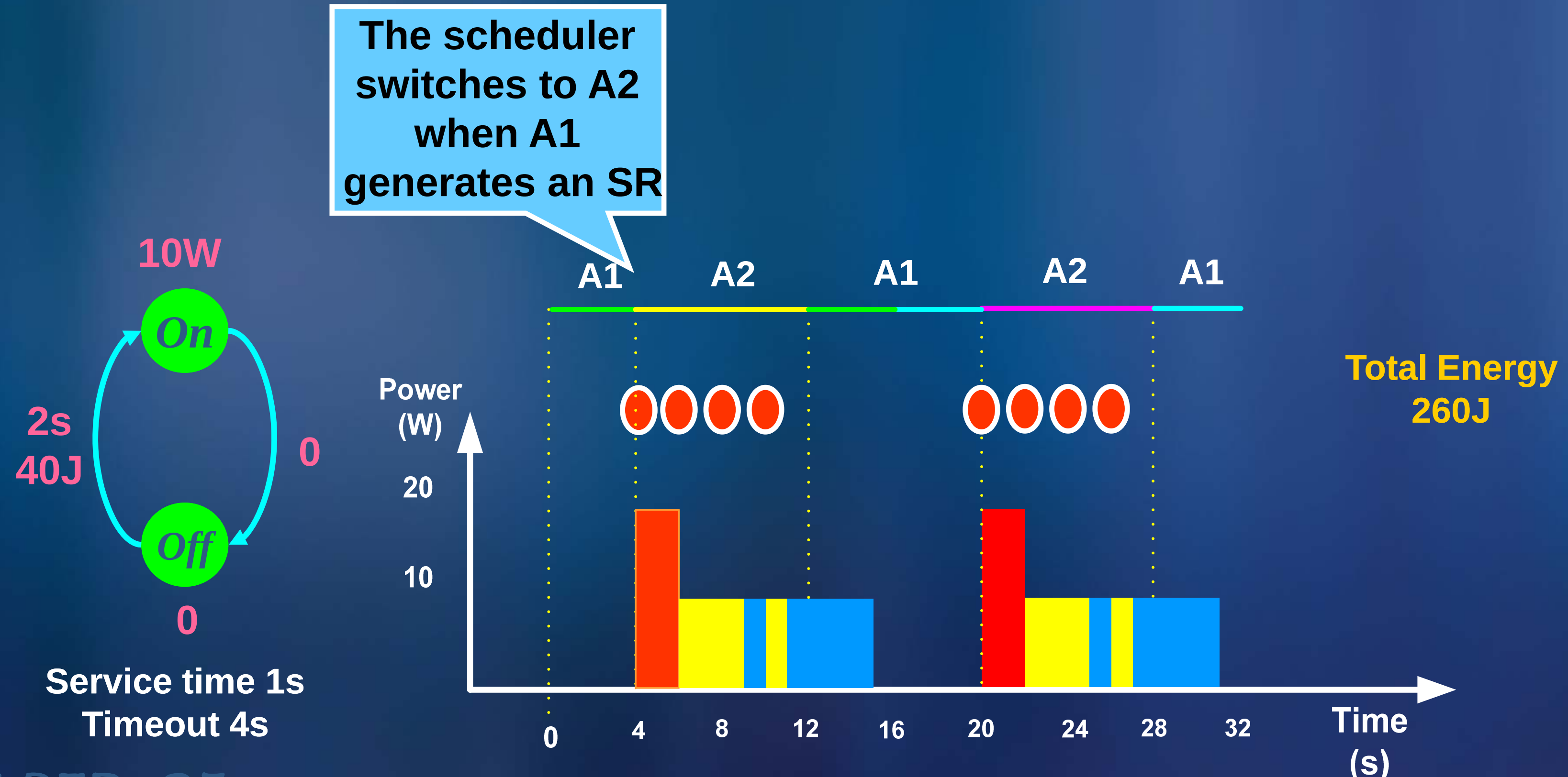




SLPED-05

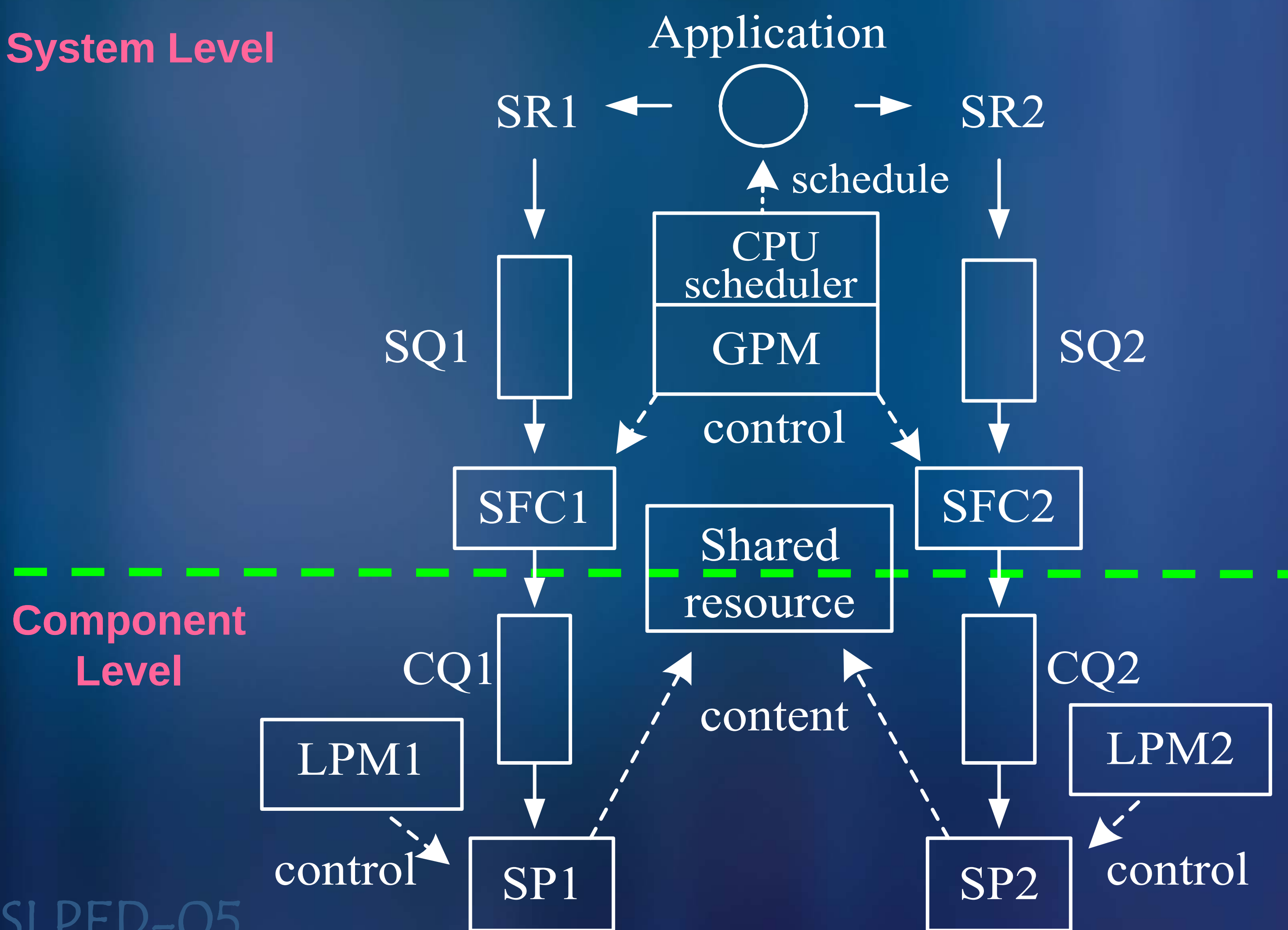
Example Cont'd

- With application scheduling
  - Scheduling is based on the number of waiting requests in the SQ



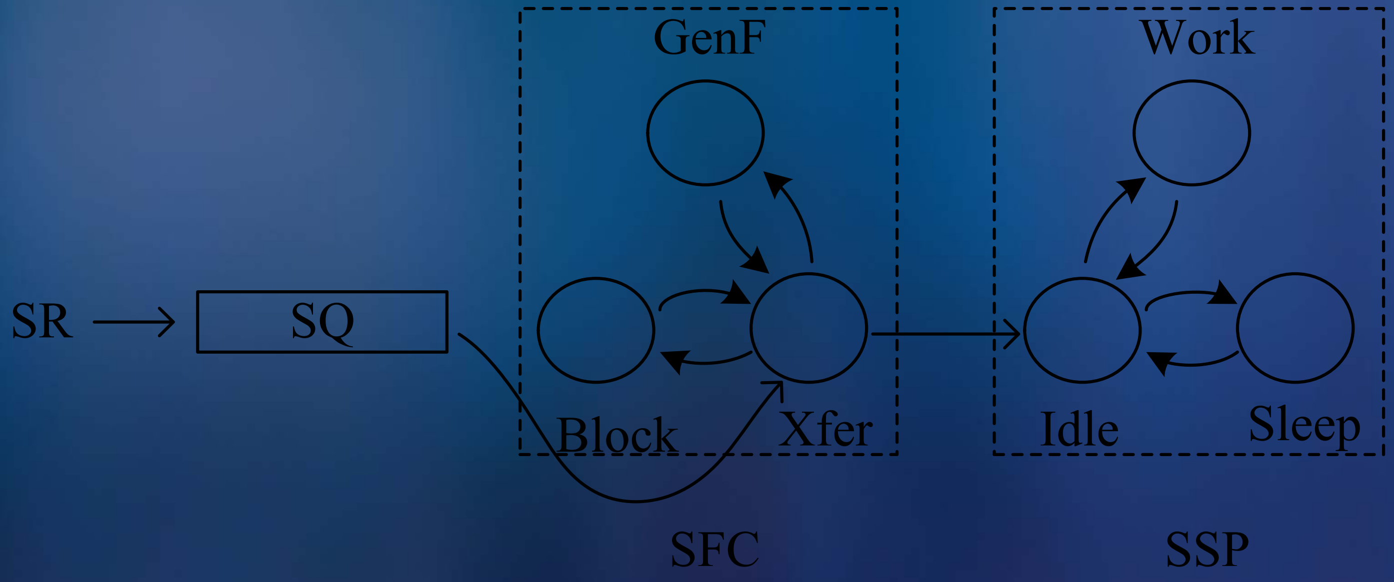
SLPED-05

CTMDP Model of HPM Architecture



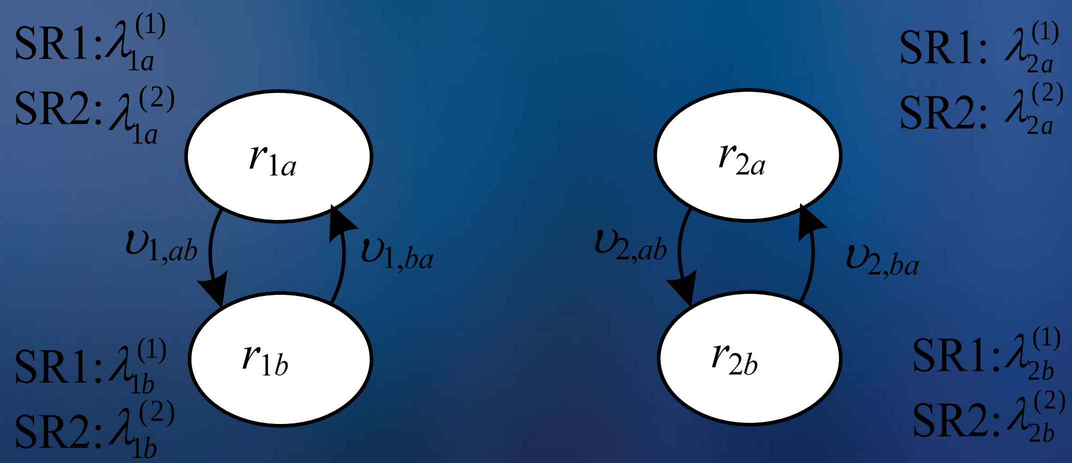
Model of Service Flow Control

- Service flow control model contains three states:
  - GenF: Generate a fake service request
  - Block: Block all incoming SR's from entering the CQ of the SP
  - Xfer: Move the SR's from the SQ to the CQ



Model of the Application

- Each application is modeled by a stationary CTMDP
  - model describes SR generation rates of the application during its execution
  - state: (r_{1a}, r_{2b})
    - r_{nx} denotes the service generation state x of application type n

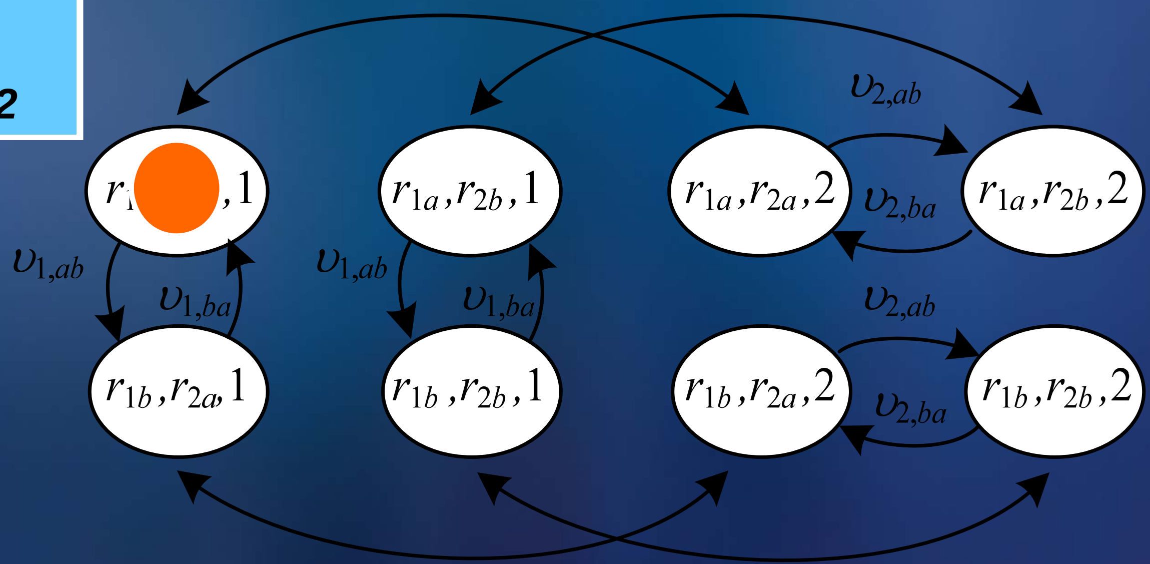


Application Type 1 Application Type 2

Model of the Application Pool

- The application pool is modeled as a stationary CTMDP
  - global states ($r_{1x}, r_{2y}, flag$)
    - *Flag* = *i* means that application *i* is running

Act:
run_Appl$_2$

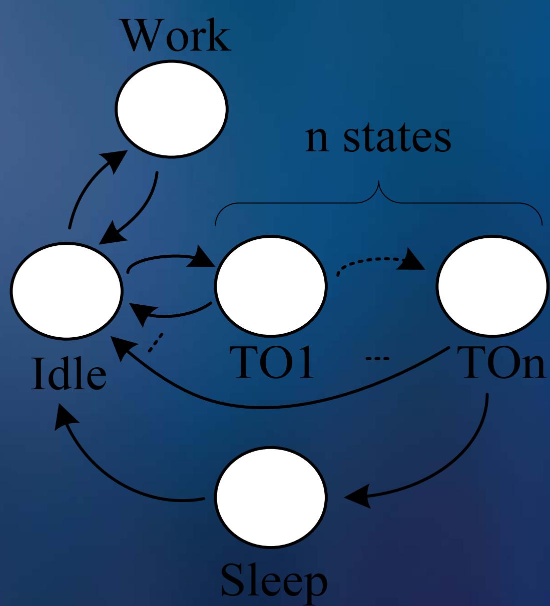


Fairness of Application Execution

- Allocate a fair share of the CPU time to each application
  - Do not intervene in the scheduling of applications that have the same workload characteristics
  - Only apply to applications that exhibit different workload characteristics
- A fairness constraint
  - Application type *i* cannot, on average, occupy more than c_i percent of the CPU time

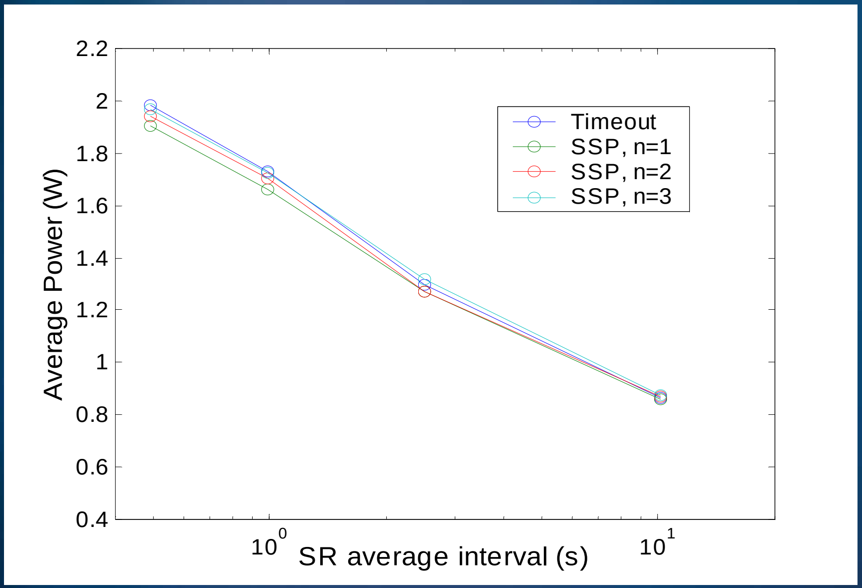
Model of Simulated Service Provider

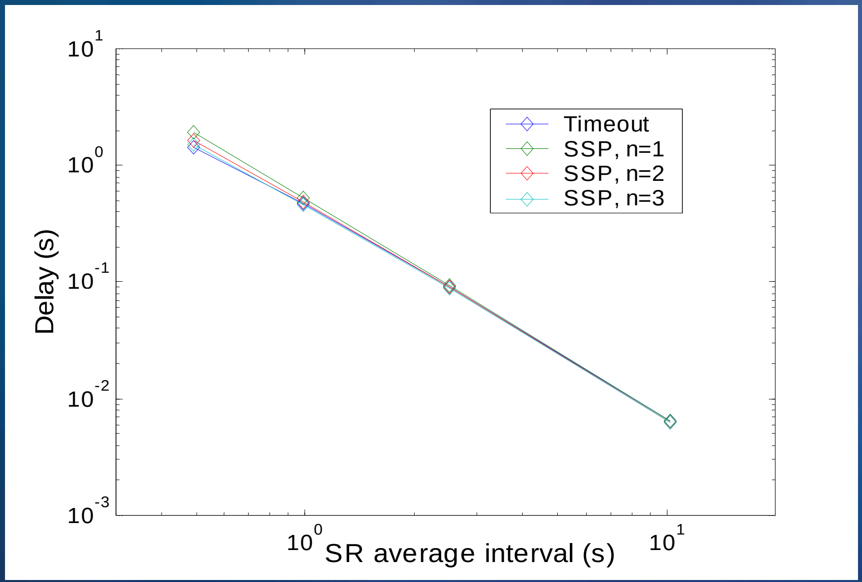
- Model of the simulated service provider
  - TO_i : non-functional time-out states, simulate the timeout policy
  - SSP autonomously transfers to TO_i state when it is idle



Verifying the SSP Model

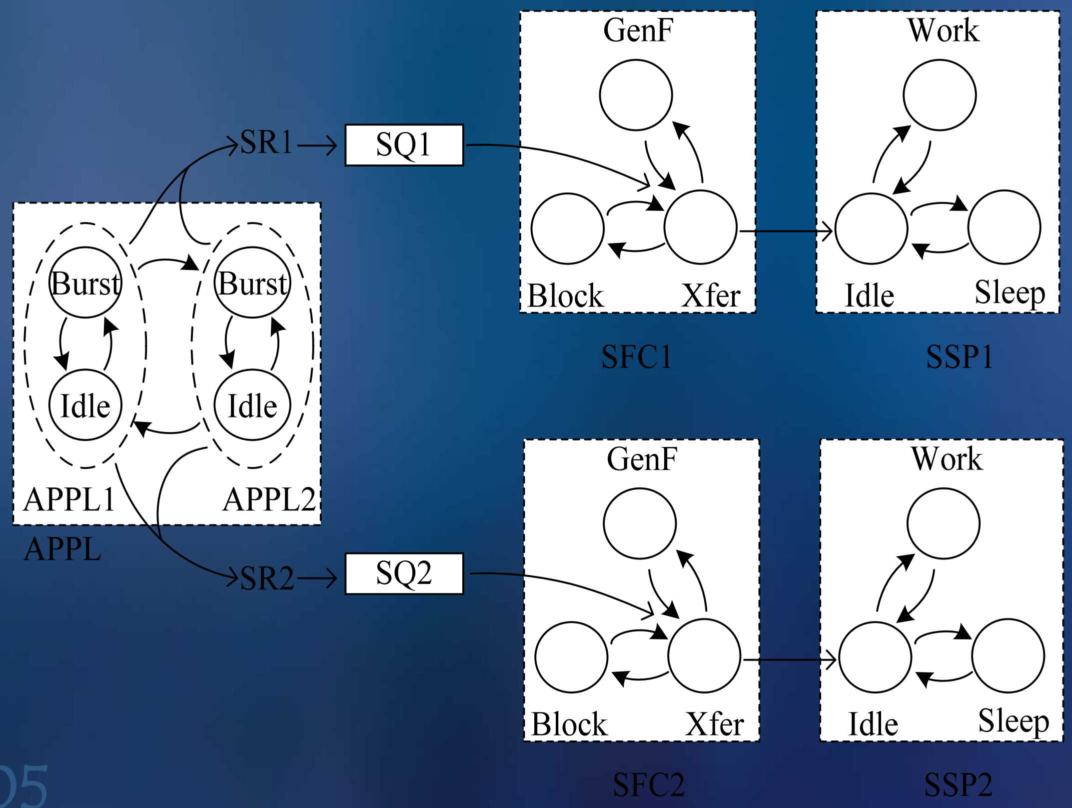
- SSP with three TO_i states provides sufficient accuracy





The Complete Model

- CTMDP-based system model
  - Components: application pool (APPL), service flow control (SFC), and simulated service provider (SSP)



Hierarchical DPM Policy Optimization

- Hierarchical DPM policy optimization is formulated as a linear programming problem

$$\text{Minimize}_{\{f_x^{a_x}\}} \left(\sum_x \sum_{a_x} f_x^{a_x} . \gamma_x^{a_x}\right)$$

$$\gamma_x^{a_x} = \tau_x^{a_x}\, pow(x, a_x) + \sum_{x' \neq x} p_{x,x'}^{a_x}\, ene(x, x')$$

$$\text{subject to: } \sum_{a_x} f_x^{a_x} - \sum_{x' \neq x} \sum_{a_{x'}} f_{x'}^{a_{x'}} . p_{xx'}^{a_{x'}} = 0, \quad \forall x \in X$$

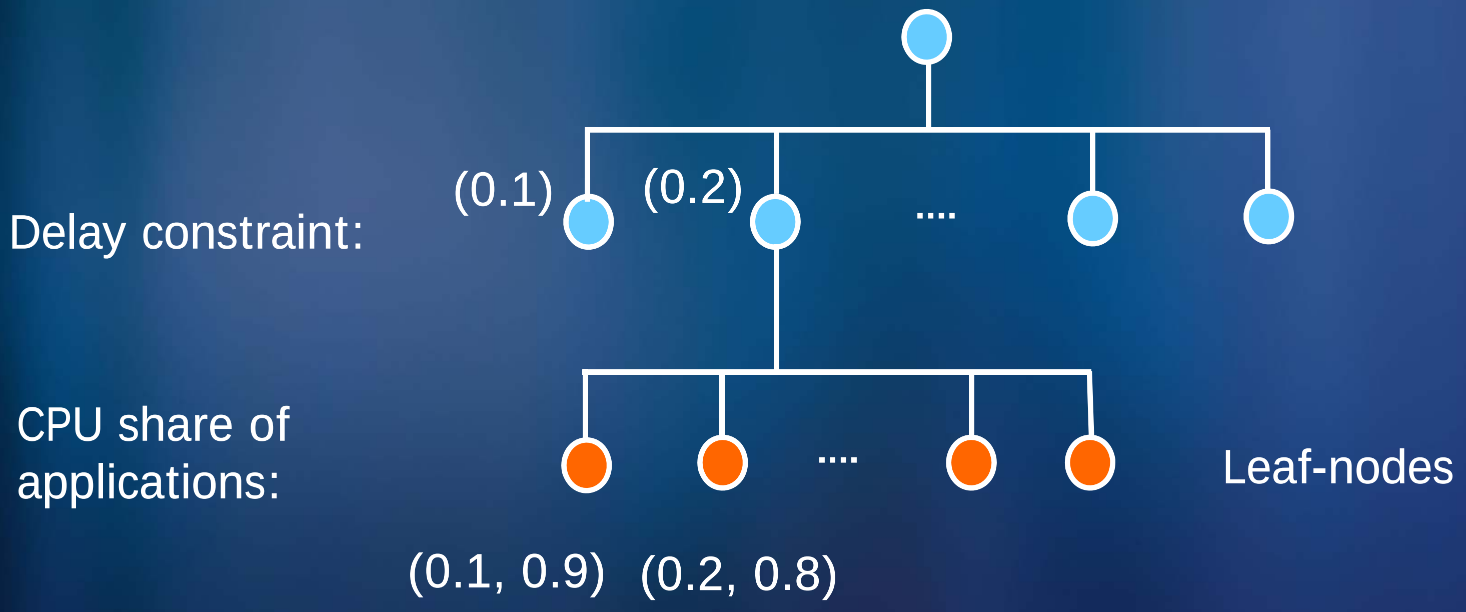
$$\sum_x \sum_{a_x} f_x^{a_x} . \tau_x^{a_x} = 1 \qquad f_x^{a_x} \geq 0$$

$$\sum_x \sum_{a_x} f_x^{a_x} \tau_x^{a_x} (q_{i,x} - D_i \lambda_{i,x}) \leq 0, \quad i = 1, 2, \ldots, I$$

$$\sum_{x:\, flag(r_x) = i} \; \sum_{a_{r,j} \in a_x} f_x^{a_x} \tau_x^{a_x} \leq c_j \times 100\%, \quad j = 1, 2, \ldots, J$$

DPM Policy Implementation

- Policy Implementation Tree
  - Each leaf-node represents a policy for a given set of system parameters, e.g., an overall delay constraint and CPU time share for different applications



Experimental Setup

- We recorded device generation traces for two types of applications: network search and file manipulation
- We used the following SR generation characteristics
  - Appl1: A Poisson process with an average rate of 0.208 requests per second
  - Appl2: A two-state CTMDP model
    - state transition rate

$$\begin{bmatrix} 0 & 0.0415 \\ 0.0063 & 0 \end{bmatrix} \left(s^{-1}\right)$$

    - SR generation rates: λ_{hd} to hard disk, λ_{wlan} to WLAN card are

$$\lambda_{hd} = [0.0826, 0.0187] \quad \lambda_{wlan} = [0.1124, 0.1124] \ \ (s^{-1})$$

Experimental Setup

- Energy and transition data of hard disk and WLAN card

Hitachi 7K60	State	Power (w)	Start-up Energy (J)	Wake-up Time (s)
	Active	2.5	--	--
	Performance idle	2.0	0	0
	Low power idle	0.85	1.86	0.4
	Stand-by	0.25	10.5	2
Orinoco WLAN	Transfer	1.4	--	--
	Receive	0.9	--	--
	Sleep	0.05	0.15	0.12

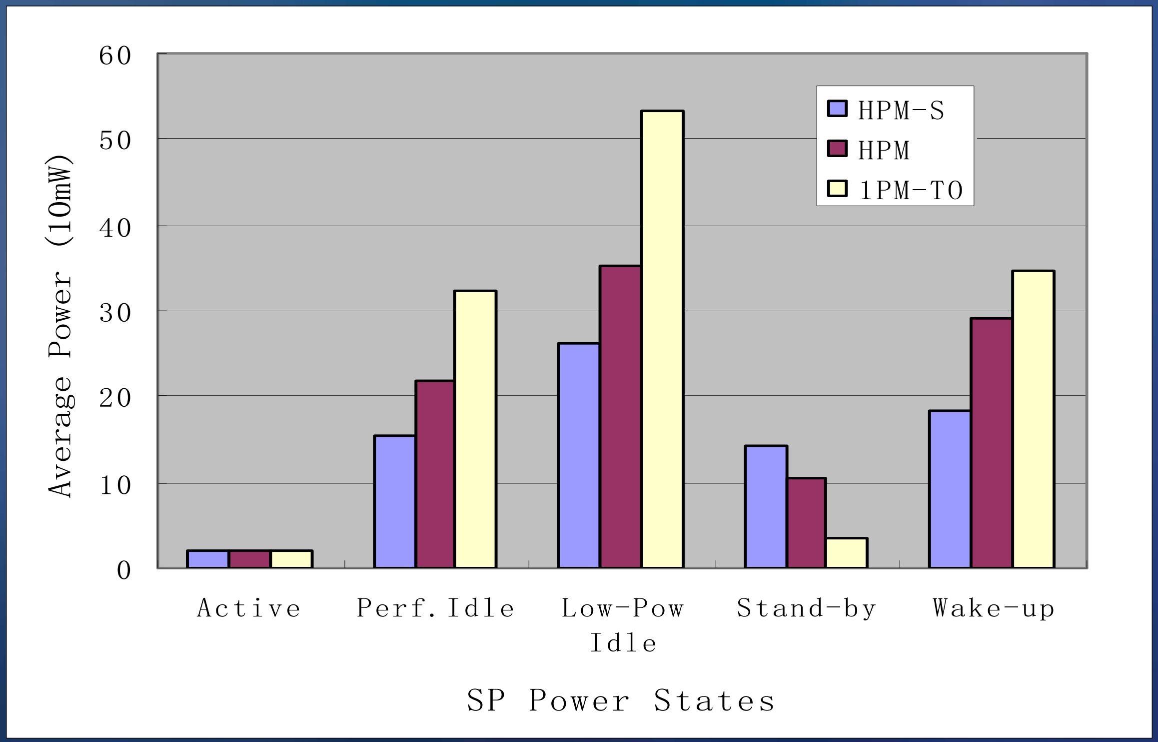
Simulation Results

- Results of Hierarchical DPM for single SP: Hard disk

CPU usage	LPM policy	Perf. Cons.	**1PM-TO (W)**	1PM-CTMDP (W)	HPM (W)	**HPM-S (W)**
0.53:0.47	TO1	0.0765	**1.2728**	1.0467	1.2591	**0.9505**
		0.5	**1.2728**	0.9309	1.0943	**0.788**
	TO2	0.0882	**1.1582**	1.0414	1.1436	**0.8651**
		0.5	**1.1582**	0.9309	1.0106	**0.7274**
0.7:0.3	TO1	0.078	**1.3805**	1.1152	1.342	**0.9951**
		0.5	**1.3805**	0.9956	1.1047	**0.8302**
	TO2	0.0903	**1.2559**	1.1107	1.2032	**1.0594**
		0.5	**1.2559**	0.9956	1.0966	**0.8734**
0.3:0.7	TO1	0.0685	**1.19**	0.9647	1.1058	**0.957**
		0.5	**1.19**	0.7922	0.9276	**0.788**
	TO2	0.076	**1.0162**	0.9451	1.012	**0.7373**
		0.5	**1.0162**	0.7922	0.8422	**0.6015**

Results Cont'd

- Distribution of average power
  - Setup (CPU usage: 0.53:0.47; Perf. Constraint: 0.5; TO1)



Results Cont'd

- Simulation results of Hierarchical DPM for both SPs: Hard disk and WLAN card

	Perf. Cons. for different SPs		**1PM - TO2 (W)**	1PM - CTMDP (W)	HPM (W)	**HPM-S (W)**
Sim1	HD	0.09	**1.157**	1.045	1.142	**0.881**
	WLAN	0.05	**0.384**	0.343	0.378	**0.310**
Sim2	HD	0.2	**1.157**	1.01	1.066	**0.788**
	WLAN	0.2	**0.384**	0.322	0.331	**0.282**

Conclusions

- A hierarchical power management architecture was proposed which aimed at facilitating power-awareness in a system with multiple components
- The proposed architecture divided power management function into two layers: system-level and component-level
- The system-level power management was formulated as a concurrent service request flow regulation and application scheduling problem
- Future Directions
  - Tune parameters of local DPM policy
  - Develop an online adaptive policy w/ variable parameters

Model of Component Dependencies

- Mutual Exclusion
  - Example: Two SPs contend for the same non-sharable resource, e.g., a low speed I/O bus
  - This type of *hard* dependence constraint can be accounted for by marking any system state that violates the mutual exclusion as invalid and by forbidding all state-action pairs that cause the system to transit to an invalid state
- Shared Resource Constraint
  - Example: SPs may want to buffer their SRs in a shared buffering area of finite size
  - This type of *soft* dependency constraint is handled by adding appropriate constraints to the system-level power optimization problem formulation