Stochastic Modeling of a Power-Managed System: Construction and Optimization

Qinru Qiu and Massoud Pedram
Dept. of EE-Systems
University of Southern California

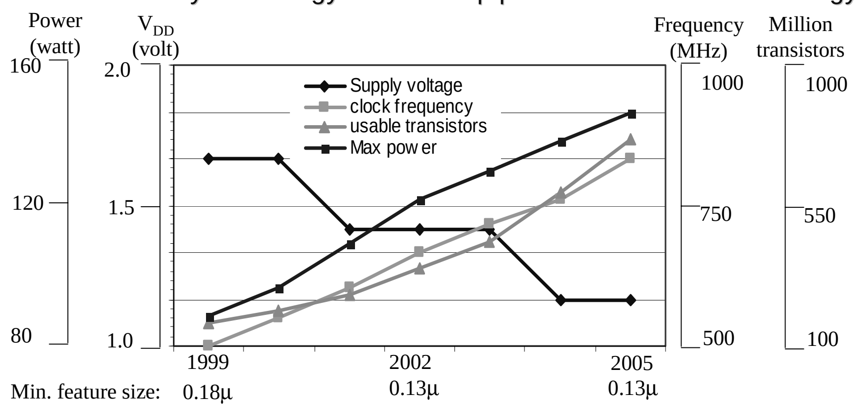
November, 2000

Outline

- Background
  - Power optimization techniques
  - Dynamic power management
- Simple systems
  - Continuous Time Markov Decision Process (CTMDP)
  - Model construction and optimization
  - Experimental results
- Complex systems
  - Generalized Stochastic Petri Nets (GSPN)
  - Model construction and optimization
  - Experimental results
- Conclusions

Motivation

- Power has become a major consideration in VLSI design
  - Power consumption will increase significantly in next few years
  - High power consumption increases the packaging and cooling cost and decreases the system reliability
  - The battery technology cannot keep pace with the VLSI technology



Information extracted from NTRS'99

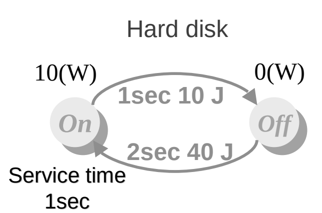
Power Saving Techniques

- Voltage and process scaling
- Low *k* dielectric and copper interconnect
- Power-aware compiler and architecture design
- Power control and management techniques
- Dynamic voltage and frequency scaling based on workload
- Better cell library design and resizing methods
- Circuit design techniques
- Low power-driven bus encoding techniques
- Low power design methodologies
- Power-conscious synthesis and design tools

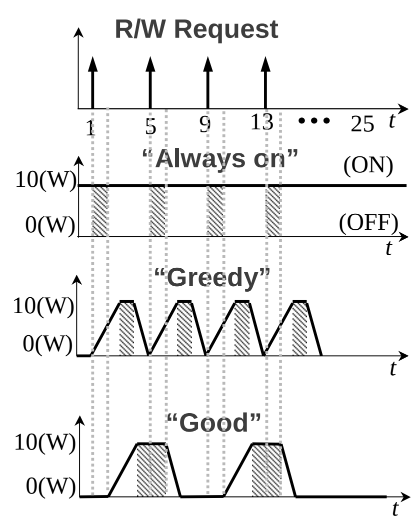
Dynamic Power Management (DPM)

- It is a system level power optimization technique
- DPM causes transitions between the system power modes to reduce power or energy dissipation while meeting the performance constraints
- Idle or under-utilized components can be shut down or slow down
- Policy refers to the type and timing of the power mode transition. Finding an optimal power management policy is a complex problem even for a simple system

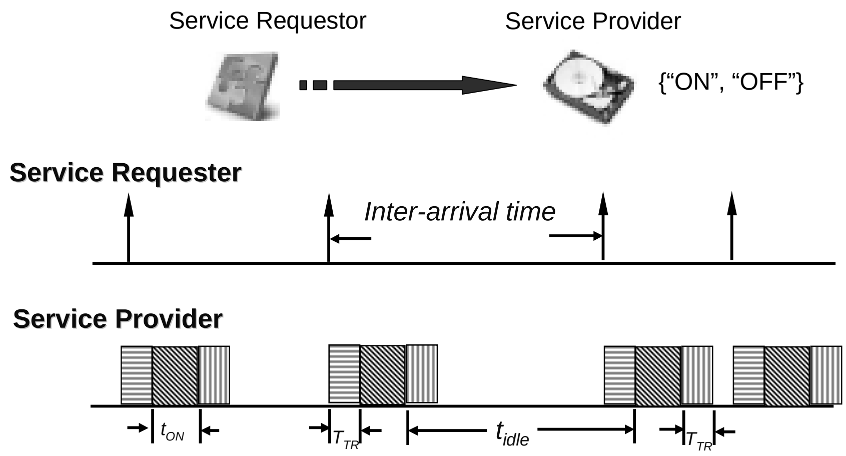
A Simple Example of DPM



	Energy	latency
Always On	**250J**	**1 sec**
Greedy	**240J**	**3 sec**
Good	**140J**	**2.5 sec**



Heuristic Approaches



Heuristic Policies

- Greedy policy
  - Turn on the server when request comes
  - Turn off the server when it is idle
  - Does not consider switching penalty
- Time-out policy
  - Turn on the server when request comes
  - Turn off when the server has been idle for $T_{threshold}$
  - No formal way to decide optimal $T_{threshold}$
  - Waste power during time-out
  - Performance penalty of wake up

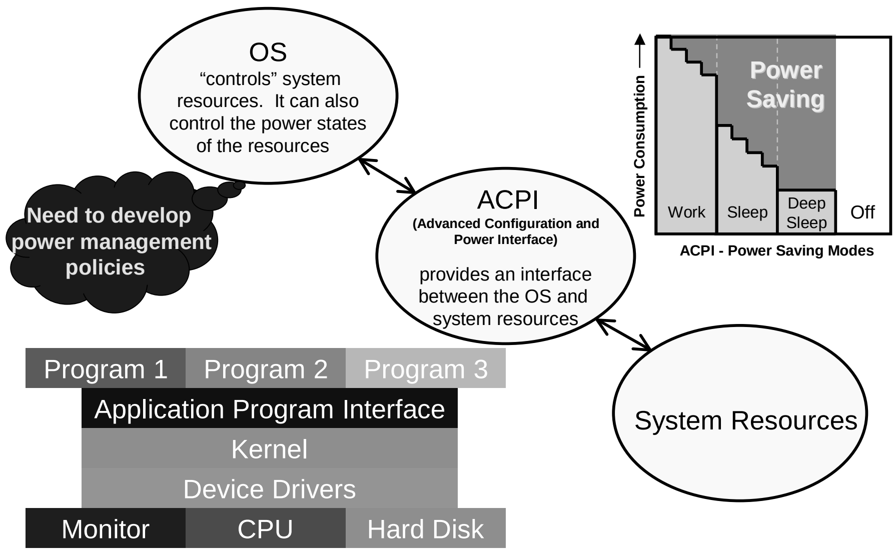
Predictive Policy

- Srivastva, Chandrakasan, et, al. 1996
  - Predict t_{idle} based on the history
  - Regression analysis based predictor
  - If $t'_{idle}[i] > T_{threshold}$, turn off the device.
  - Turn on the device as soon as request comes
- C.H. Huang, et. al. 1997
  - Pre-wakeup the device after it has been idle for t'_{idle}
  - Reduces timing penalty in wake up, but consumes more power

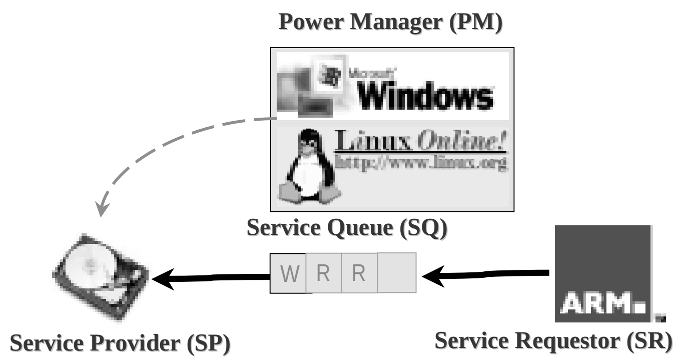
Stochastic Based Approach

- DPM based on Discrete Time Markov Decision Process (DTMDP) by L. Benini et. al., 1998
  - The system is modeled as DTMDP
  - The optimal policy is obtained using Linear Programming (LP)
  - Significant improvement in theoretical framework
  - Limitations:
    - Some assumptions are not practical
    - The state transition probability is difficult to obtain
    - Power Manager (PM) needs to send control signal in every time-slice

Operating System-directed Power Management (OSPM)

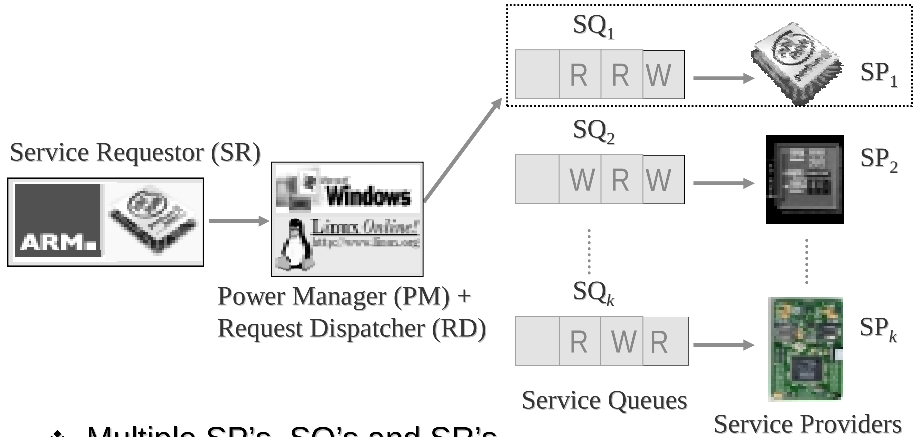


A Simple Power-managed System



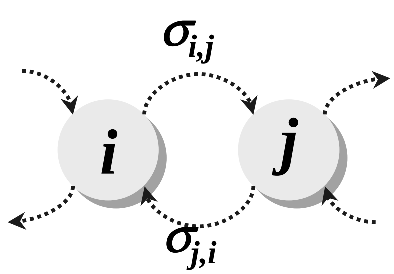
- A single SP, a SQ and a SR
- The request inter-arrival time, service time and SP switching times are assumed to follow exponential distribution
- The System is modeled as a Continuous Time Markov Decision Process (CTMDP)

A Complex Power-managed System



- Multiple SP's, SQ's and SR's
- Complex system behavior and components interaction
- The system is modeled as a Controllable Generalized Stochastic Petri Net (CGSPN)

Continuous Time Markov Process



♦***stochastic process:*** **a family of random variables {*X*(*t*), *t*≥0}**

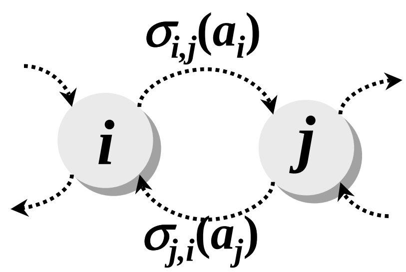
♦***Markov process:*** **a stochastic process that for any time $t_0<t_1<\ldots<t_n<t$, $P[X(t) \le x \mid X(t_n) = x_n, \ldots, X(t_0) = x_0] = P[X(t) \le x \mid X(t_n) = x_n]$**

$$\mathbf{G} = \begin{bmatrix} -\sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} & \cdots \\ \sigma_{2,1} & -\sigma_{2,2} & \sigma_{2,3} & \cdots \\ \sigma_{3,1} & \sigma_{3,2} & -\sigma_{3,3} & \cdots \\ \vdots & \vdots & \vdots & -\sigma_{N,N} \end{bmatrix}$$

$$\sigma_{i,j} = \lim_{t \to 0} \frac{p_{i \Rightarrow j}(t)}{t} = p'_{i \Rightarrow j}(0)$$

$$\sigma_{i,i} = \sum_{j \ne i} \sigma_{i,j}$$

Controllable Markov Process



$$\mathbf{G} = \begin{bmatrix} -\sigma_{1,1}(a_1) & \sigma_{1,2}(a_1) & \cdots \\ \sigma_{2,1}(a_2) & -\sigma_{2,2}(a_2) & \cdots \\ \vdots & \vdots & -\sigma_{N,N}(a_N) \end{bmatrix}$$

$\sigma_{i,j}$ can be controlled by command a_i

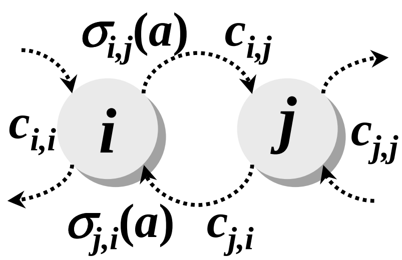
Action a_i: A command taken in state *i*

Action Set A_i: Available commands in state *i*

Deterministic vs. Randomized Policy

- Policy (π): The set of state-command pairs $<i, a(t)>$, $a(t) \in A_i$
- Deterministic policy
  - The action $a(t)$ is chosen from A_i with probability 1
- Randomized policy
  - The action $a \in A_i$ is chosen with probability $p_i^a(t)$
  - $\sum p_i^a(t) = 1$, $a \in A_i$

Controllable Markov Process With Cost



State Cost Rate:

$$c_i(a) = c_{ii} + \sum_{j \neq i} \sigma_{ij}(a) c_{ij}$$

c_{ii}: System cost per unit time if system stays in state *i*

c_{ij}: System cost if system makes a transition from state *i* to state *j*

Markov Decision Process

System Cost :

$$c_{i,avg} = \lim_{t \to \infty} \frac{1}{t} \int_0^t \sum_{j=1}^{n} p_{i \Rightarrow j}(\tau) c_j \, d\tau$$

$p_{i \Rightarrow j}(\tau)$: state probability of *j* at time τ if the system initial state is *i*

The system cost in a Markov decision process is policy dependent

Policy optimization: Find an optimal policy π such that the average cost is minimized

Stationary policy: $a_i(t)$ (or $p_i^{a_i}(t)$) is the same for all times

Theorem: A stationary policy is optimal for the Markov decision process

Constrained Markov Decision Process

- The system contains an objective cost *c_obj* and several constraint costs *c_con*
  - Definitions of *c_obj* and *c_con* are system-dependent
- Constrained policy optimization

$$\text{Minimize}_{\pi}\,(c\_obj^{\boldsymbol{\pi}}_{i,avg})$$

$$\text{such that}: c\_con^{\boldsymbol{\pi}}_{i,avg} < \text{Constraint}$$

- Theorem: If the constraint is inactive, the optimal policy is a deterministic policy, otherwise it may be a randomized policy

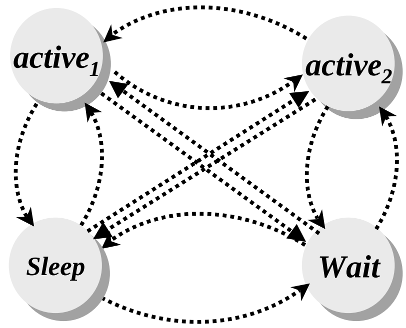
Simple DPM System Modeling: Overview

- Each component is modeled as a CTMDP
- The entire system is modeled as a composition of the individual component models
- The generator matrix of the composed model is calculated using a Tensor sum operation
- Special effort is expended to correctly handle the synchronization between SP and SQ
- The idle and busy states of the SP are separated; Transitions from busy to idle state is not controllable
- Constraints are applied to the action sets to ensure that the overall model is reasonable. This also ensures that the policy optimization problem can be solved

Service Provider (SP)

Required information: **power modes, actions, parameters**

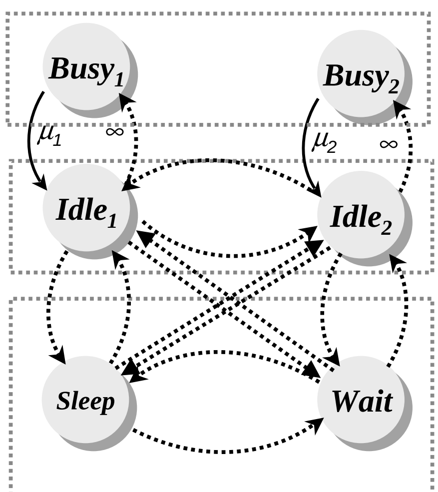
$pow(s_i)$	Power consumption
$\mu(s_i)$	Average service speed
$1/\chi_{s_i,s_j}$	Average transition time
$ene(s_i, s_j)$	Energy cost



S={*active*$_1$, *active*$_2$, *wait*, *sleep*}

A={*go_active*$_1$, *go_active*$_2$, *go_wait*, *go_sleep*}

Busy state vs. Idle State

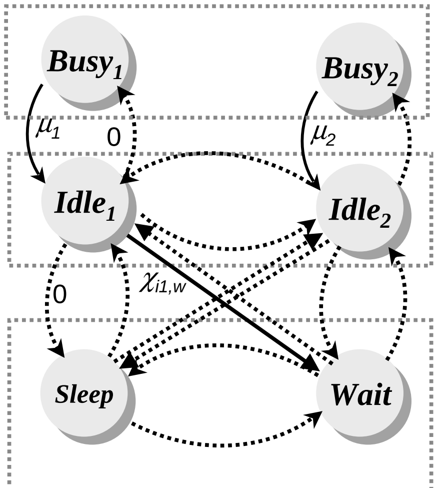


States: *busy*, *idle*, *power down*
(*busy*, *idle*: active state)

busy → corresponding *idle*
Idle → *power down* or *idle* or corresponding *busy*

$\chi_{busy,\ idle} = \mu$
$\chi_{idle,\ busy} = \infty$
$\chi_{idle,\ power\ down} = \chi_{a,\ power\ down}$

Generator Matrix of SP



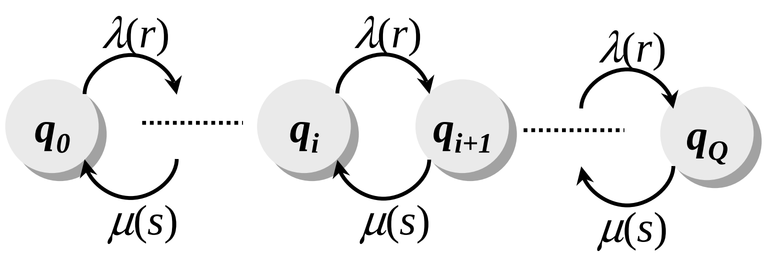
The parametric generator matrix $G_{SP}(a)$ is a function of *a* (*action*)

$$\sigma_{s_i,s_j}(a) = \delta(s_j, a) \cdot \chi_{s_i,s_j},\ s_i \neq s_j$$

$$\sigma_{s_i,s_i}(a) = -\sum_{s_j \neq s_i} \sigma_{s_i,s_j}(a)$$

$$\delta(s, a) = \begin{cases} 1 & s \text{ is the destination state of action } a \\ 0 & \text{otherwise} \end{cases}$$

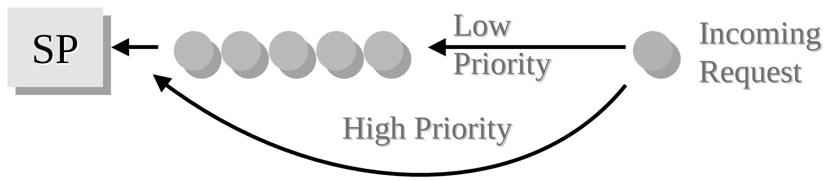
Single Service Queue (SSQ)



- Shortcomings
  - Assumes all requests have the same priority, which is not true in general
  - Can only use one delay constraint, which is not flexible enough to handle different types of requests

Priority Service Queue (PSQ)

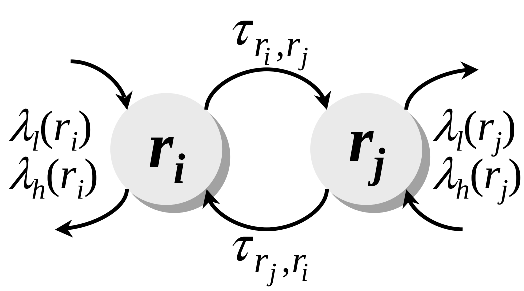
A simple priority queue in OS



Abstract model

Correlation: Request in LSQ can be serviced only when there is no request in HSQ

Service Requester (SR)



System Model

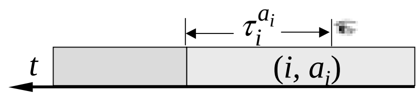
- The system (SYS) can be modeled as the composition of the Markov processes of SR, SP and SQ
  - state set: $X = S \times Q \times R$ - {invalid states where SP is busy and SQ is empty}
  - generator matrix $G_{SYS}(a)$ gives the state transition rates under action a
  - Action set: A_x for each state x

Policy Optimization

- Linear Programming
  - Optimal randomized policy (global optimal)
- Non-linear Programming
  - Optimal deterministic policy
- Branch & Bound Algorithm
  - Optimal deterministic policy
- Policy Iteration
  - Modification of conventional unconstrained optimization algorithm
  - Only finds optimal deterministic policy with certain property

Some Variables to Measure CTMDP

- $\tau_i^{a_i}$ **: expectation of the time that the system will be in state *i* and a_i is chosen** $\tau_i^{a_i} = 1/\sum_{j\neq i}\sigma_{ij}^{a_i}$
- $x_i^{a_i}$ **: frequency that the state is *i* and action a_i is taken (*state action prob*** $x_{a_i}\cdot\tau_i^{a_i}$ **)**
- $p_{ij}^{a_i}$ **: probability that the next system state is *j* if current state is *i* and action a_i is taken** $p_{ij}^{a_i} = \sigma_{ij}^{a_i} / \sum_{l\neq i}\sigma_{il}^{a_i}$
- $\gamma_i^{a_i}$ **: expected cost during the time the system stays in state *i* and a_i is taken** $\gamma_i^{a_i} = c_{ii}\tau_i^{a_i} + \sum_{j\neq i} c_{ij}p_{ij}^{a_i}$



Calculate Variables In Our System

- Three different $\gamma_i^{a_i}$

$$c\_pow_i^{a_i} = c\_pow_{ii}\tau_i^{a_i} + \sum_{j\neq i} c\_pow_{ij}p_{ij}^{a_i}$$

$$c\_lsq_i^{a_i} = c\_lsq_{ii}\tau_i^{a_i}$$

$$c\_hsq_i^{a_i} = c\_hsq_{ii}\tau_i^{a_i}$$

LP Based Optimization

$$\text{Minimize}_{\{x_i^{a_i}\}} \ (\sum_i \sum_{a_i} x_i^{a_i} c\_pow_i^{a_i})$$

Min. C_obj

$$\text{subject to}: \sum_{a_i} x_i^{a_i} - \sum_j \sum_k x_j^{a_j} p_{ji}^{a_j} = 0, i \in X$$

$$\sum_i \sum_{a_i} x_i^{a_i} \tau_i^{a_i} = 1$$

$$x_i^{a_i} \geq 0$$

Property of CTMDP

$$\sum_i \sum_{a_i} x_i^{a_i} c\_hsq_i^{a_i} < D_H$$

$$\sum_i \sum_{a_i} x_i^{a_i} c\_lsq_i^{a_i} < D_L$$

Performance constraints

- Only gives the randomized policy

NLP Based optimization

- A NLP based optimization approach is used to find the optimal deterministic policy
  - Deterministic policy: for each state *i*, there is only one a_i that $x_i^{a_i} \neq 0$
  - Not an ILP because $x_i^{a_i}$ may not be an integer
  - $x_i^a \cdot x_i^{a'} = 0, a, a' \in \boldsymbol{A_i}, a \neq a'$, $\Sigma_i A_i(A_i\text{-}1)/2$ more constraint

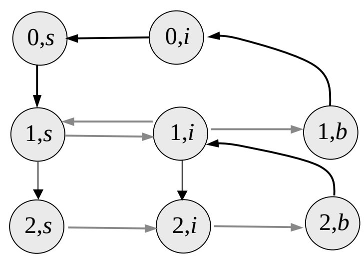
$$\text{Minimize}_{\{x_i^{a_i}\}} \ (\sum_i \sum_{a_i} x_i^{a_i} c\_pow_i^{a_i})$$

⇩

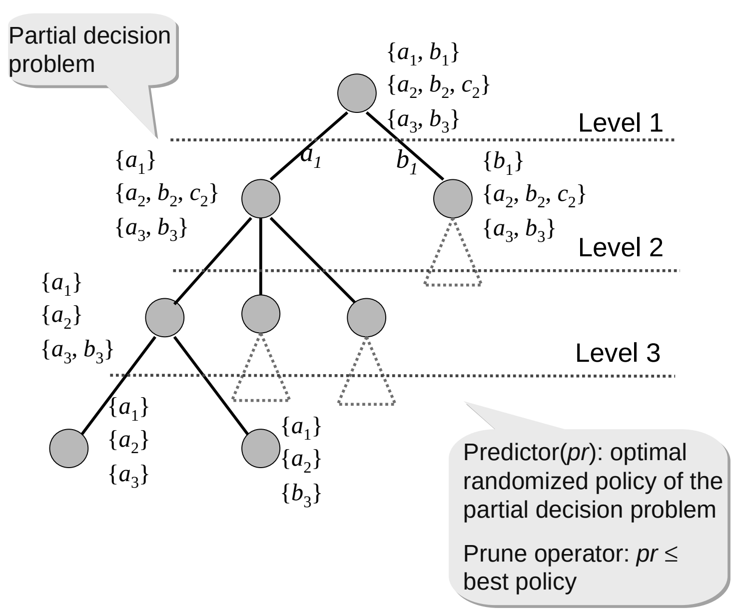
$$\text{Minimize}_{\{x_i^{a_i}\}} \ (\lambda \sum_i \sum_{a_i \neq l, a_i, l \in A_i} x_i^{a_i} \cdot x_i^l + \sum_i \sum_{a_i} x_i^{a_i} c\_pow_i^{a_i})$$

Motivation for Branch-Bound

- Branch-and-bound is used to solve the ILP
- Decision in each state has a significant impact on the system performance and power consumption
  - Prune inefficient policies early on



Decision Tree for Brand-and-Bound



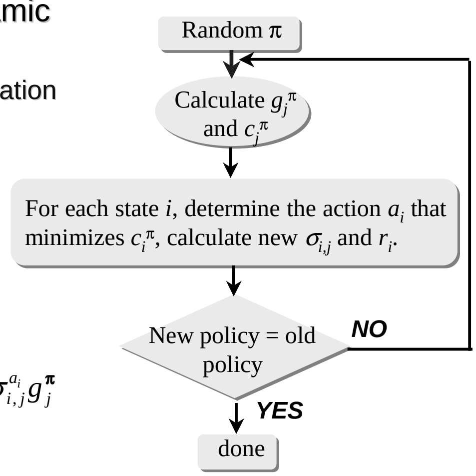
Policy Iteration

- Policy iteration: dynamic programming
  - Unconstraint optimization

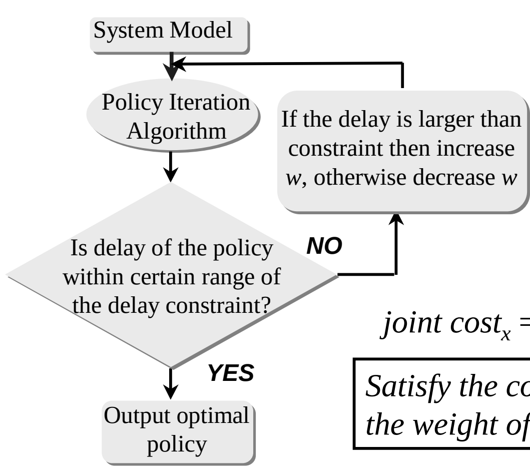
$$\sum_{j=1}^{N} \sigma_{i,j}^{a_i} g_j^{\pi} = 0$$

$$g_i^{\pi} = c_i^{a_i} + \sum_{j=1}^{N} \sigma_{i,j}^{a_i} c_j^{\pi}$$

$$c_i^{\pi}(t) = c_i^{a_i} + \sum_{j=1}^{n} \sigma_{i,j}^{a_i} c_j^{\pi} + t\sum_{j=1}^{N} \sigma_{i,j}^{a_i} g_j^{\pi}$$



Modified Policy Iteration

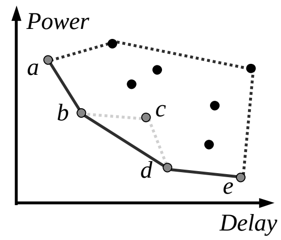


joint cost$_x$ = *c_pow*$_x$+*w*·*c_delay*$_x$

Satisfy the constraint by changing the weight of c_con in joint cost

Policy Iteration Based Optimization

- Convex policy: (p_i, d_i)
  - $\forall j,\ p_j < p_i \Rightarrow d_j > d_i$
  - $\forall j, l,\ p_j > p_i > p_l \Rightarrow (d_i - d_j)/(p_j - p_i) < (d_l - d_i)/(p_i - p_l)$



- Proposition: The output of the modified policy iteration algorithm is an optimal convex policy which satisfies the performance constraint

Experimental Results: Simple DPM System With SSQ

- System model
  - A SP with three power mode: active, sleep, standby
  - High power consumption when the SP is busy
  - When SP is active, average service time is 8ms
  - Two different distribution for SP transition time (TD)
    - Exponential distribution (Exp) & uniform distribution (Uni)
  - A SQ model with length 20

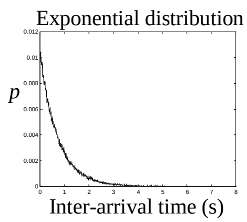
State	sleep	stdby	idle	busy
P (W)	0.13	0.35	0.95	2.15
$1/\mu$ (s)	0	0	0	0.008

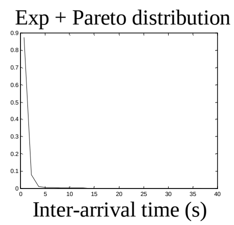
E (J)	sleep	stdby	idle
sleep	0	5.1	7.0
stby	0.006	0	2
idle	0.067	0.001	0

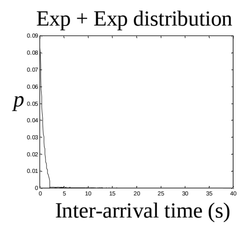
T_{tr} (s)	sleep	stdby	idle
sleep	0	0.6	1.6
stdby	0.3	0	1.2
idle	0.67	0.4	0

Input Trace

- SR has one requestor generation state
  - Average requestor inter-arrival time is 0.72 sec
  - Five different distribution (RD)
    - Exponential distribution (Exp)
    - Combination of exponential distribution & Pareto distribution (Exp&Par)
      - ***Pareto:*** **$f(t)=1-at^{-b}$**
      - **Has longer idle time than exponential distribution**
    - Combination of two exponential distribution (Exp & Exp)
    - Uniform distribution (Uni)
    - Normal distribution (Nor)





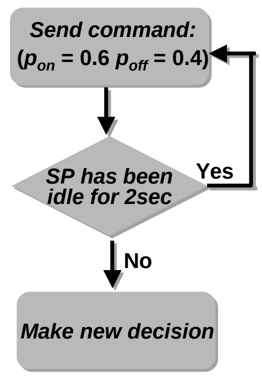


Experimental Policies

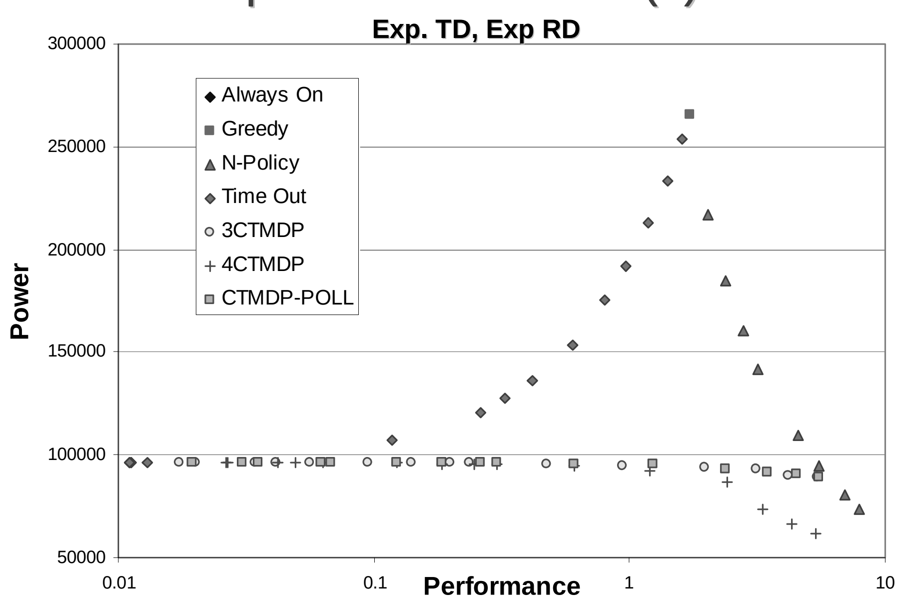
- Always on policy
  - Reasonable choice for system with high switching penalty
- Time-out policy
  - Three SP power modes: *busy, idle, sleep*
  - Vary time-out period to obtain a set of performance-power trade offs
- N-policy
  - Three SP power modes: *busy, idle, sleep*
  - turn on the server when there are *N* requests waiting and turn off the server when there are no requests
  - Optimal deterministic policy if the system has only two states
  - Vary the number N to obtain a set of performance-power trade offs

Experimental Policies (cont.)

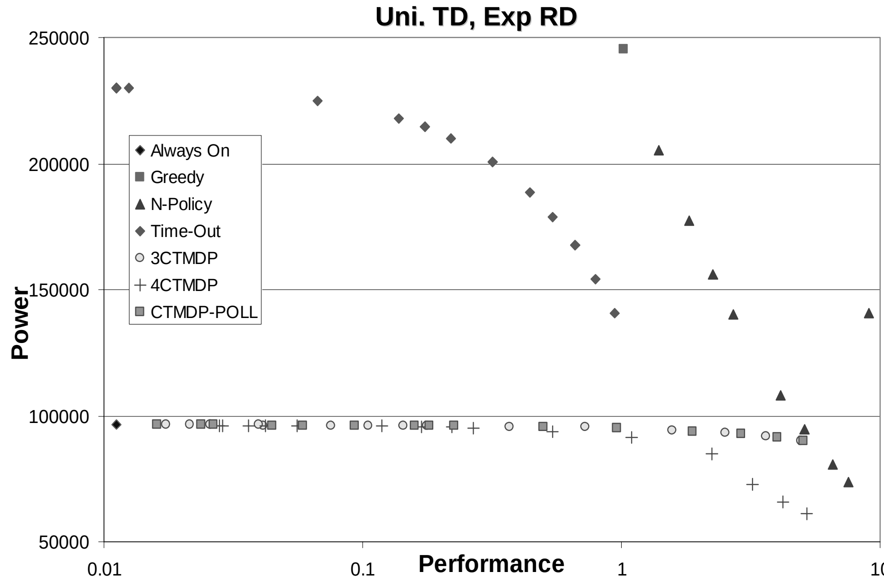
- Theoretical CTMDP policy
  - Three SP power modes
  - Four SP power modes
  - Vary performance constraint to obtain a set of performance-power trade offs
- Modified CTMDP policy: CTMDP-Poll
  - The PM will re-issue the command if the system has been idle for a long time, so that the probability for turning off is increased
  - Three SP power modes: *active, idle, sleep*
  - Vary performance constraint to obtain a set of performance-power trade offs



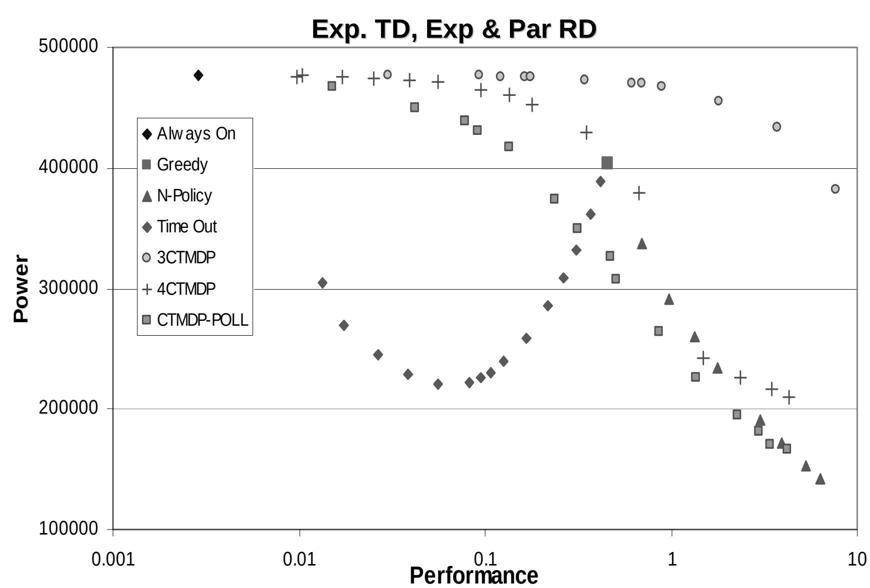
Experimental Results (1)



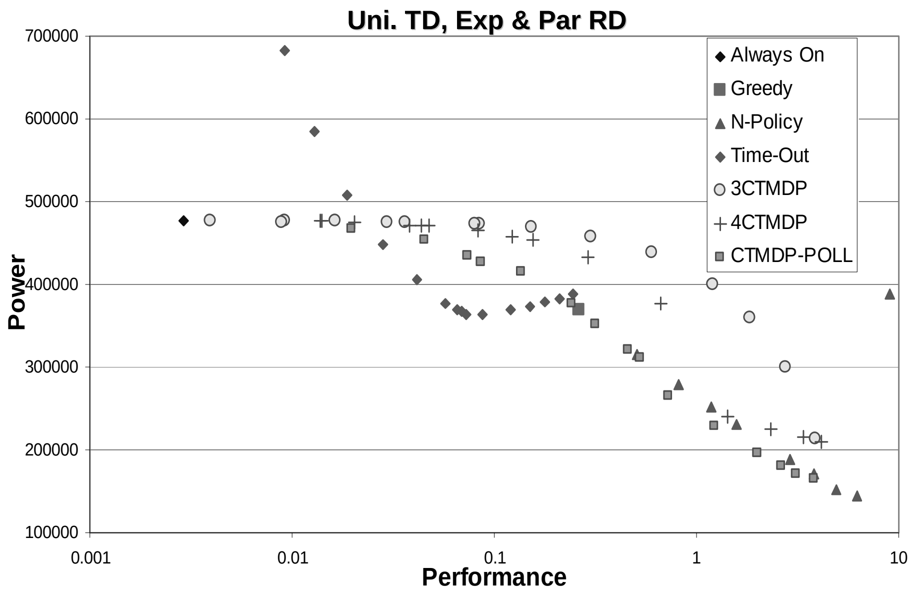
Experimental Results (2)



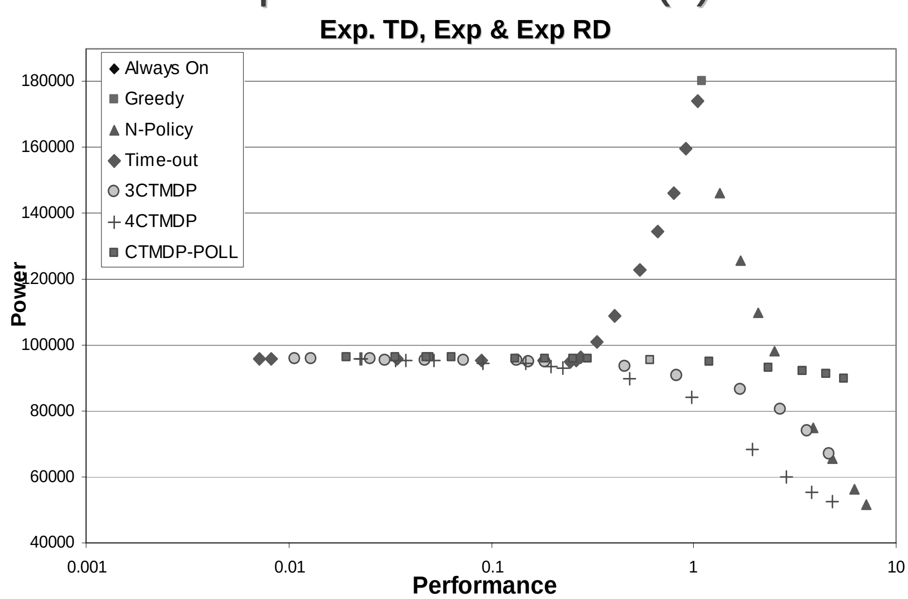
Experimental Results (3)



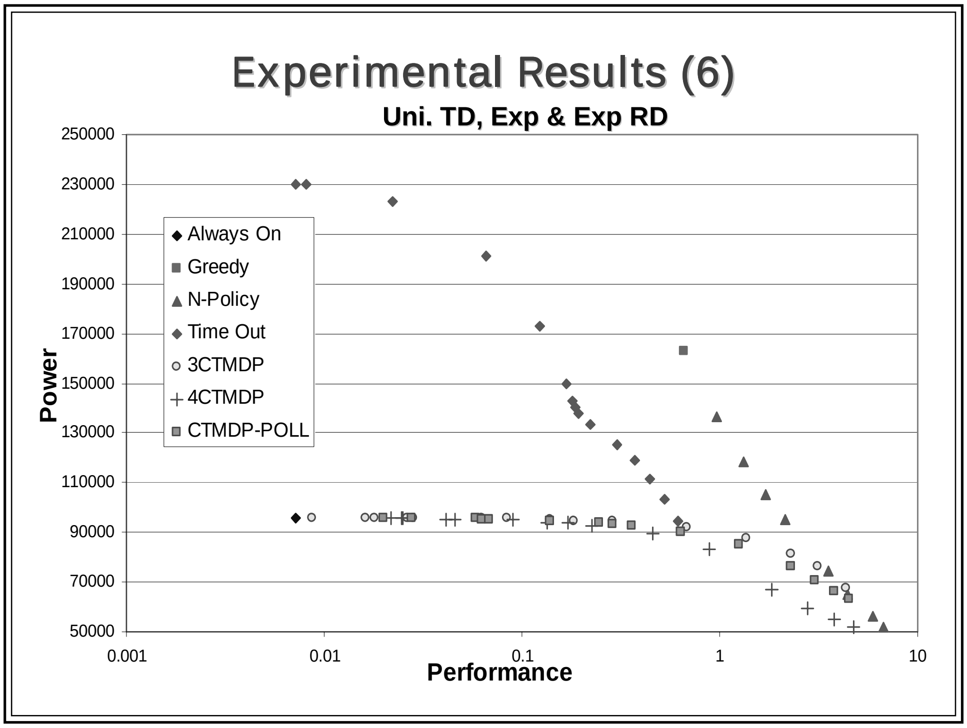
Experimental Results (4)



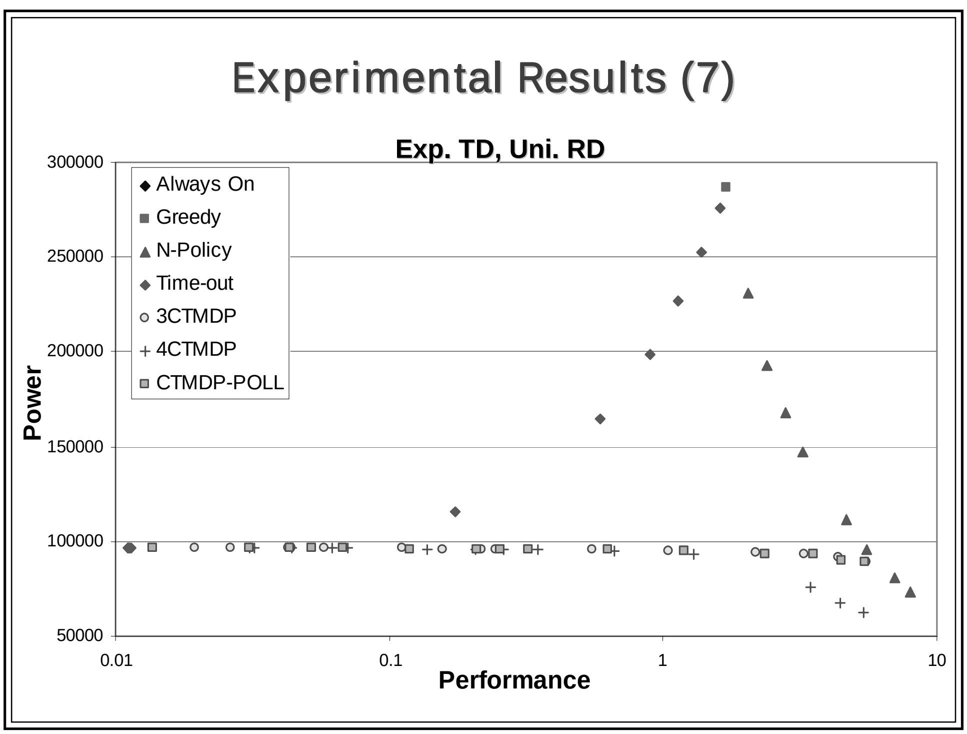
Experimental Results (5)



Page 23

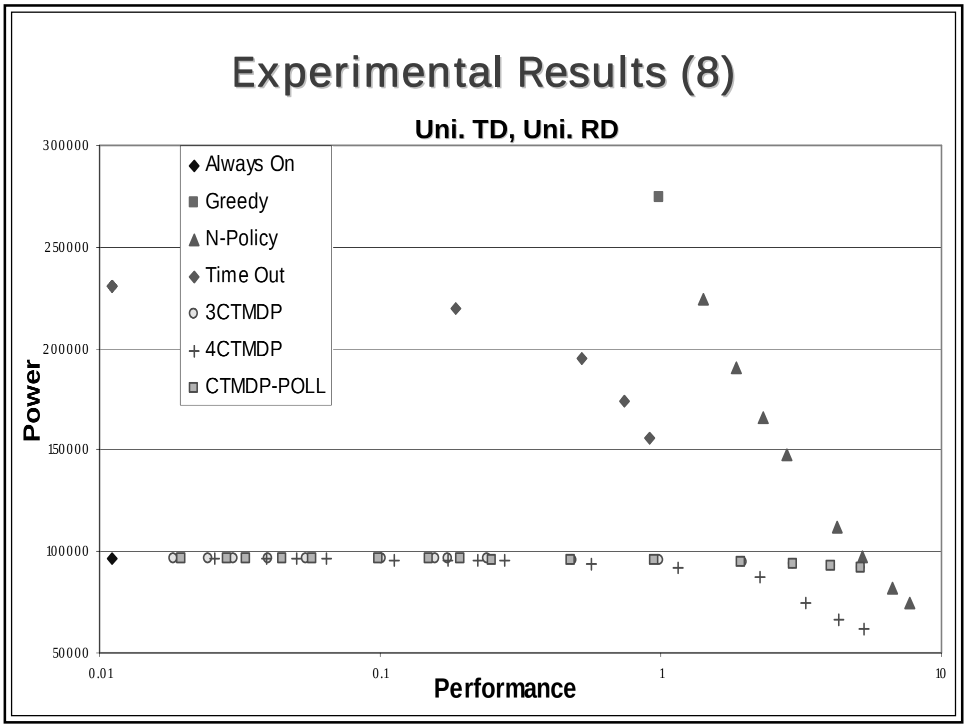


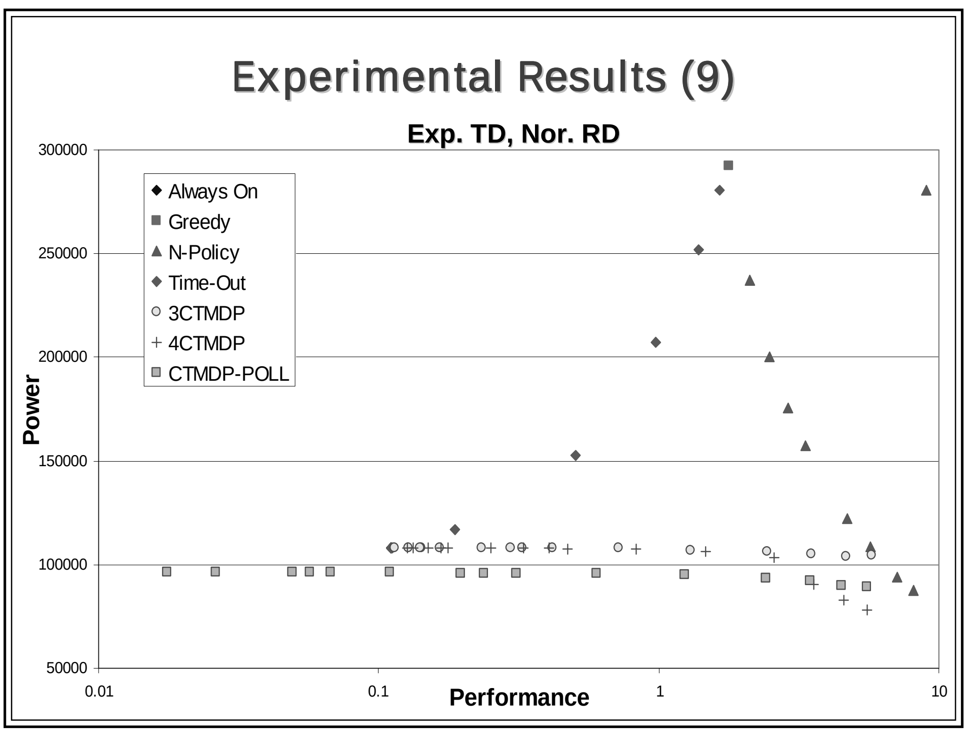
Experimental Results (6)
Uni. TD, Exp & Exp RD
Power
Performance
Always On
Greedy
N-Policy
Time Out
3CTMDP
4CTMDP
CTMDP-POLL



Experimental Results (7)
Exp. TD, Uni. RD
Power
Performance
Always On
Greedy
N-Policy
Time-out
3CTMDP
4CTMDP
CTMDP-POLL

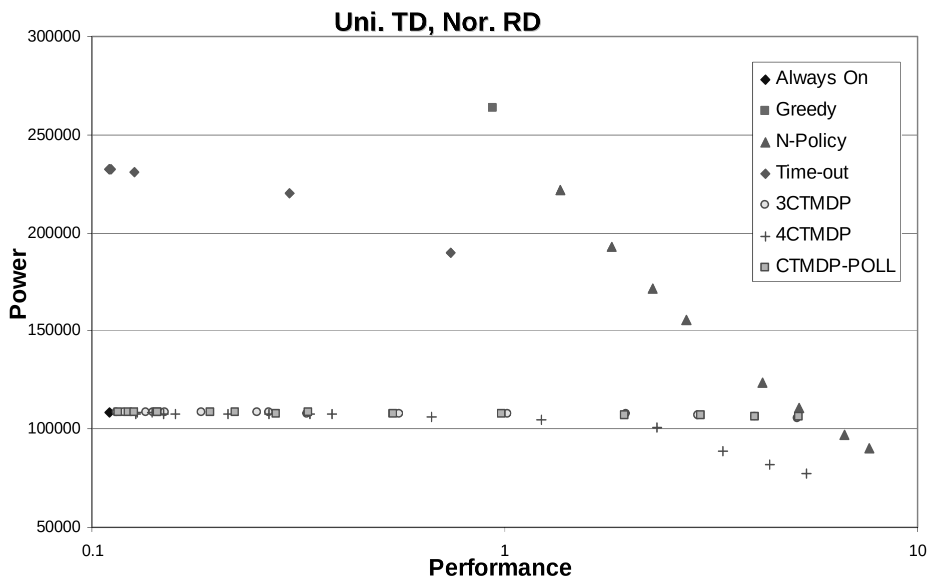
Page 24


Experimental Results (8)
Uni. TD, Uni. RD
Always On
Greedy
N-Policy
Time Out
3CTMDP
4CTMDP
CTMDP-POLL
Power
Performance
300000
250000
200000
150000
100000
50000
0.01
0.1
1
10


Experimental Results (9)
Exp. TD, Nor. RD
Always On
Greedy
N-Policy
Time-Out
3CTMDP
4CTMDP
CTMDP-POLL
Power
Performance
300000
250000
200000
150000
100000
50000
0.01
0.1
1
10

Page 25

Experimental Results (10)



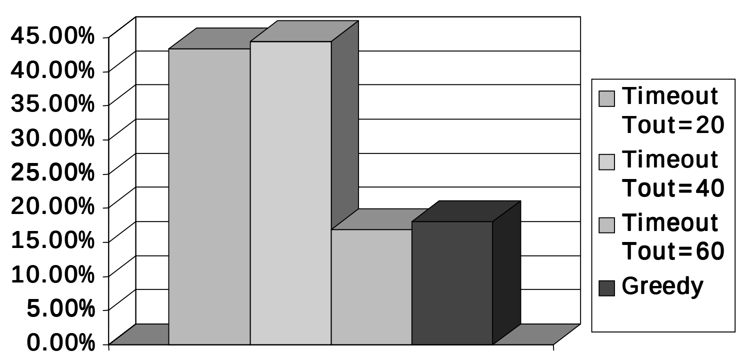
Analysis

- The stochastic policies out perform the heuristic policies
- The stochastic policies can provide power delay trade off
- Three state CTMDP policy is not efficient with input sequence with Exp & Pareto inter-arrival time
- CTMDP-poll policy solves the above problem
- Four state CTMDP is robust in different TD, RD distribution

Experimental Results: Simple DPM Systems With PSQ

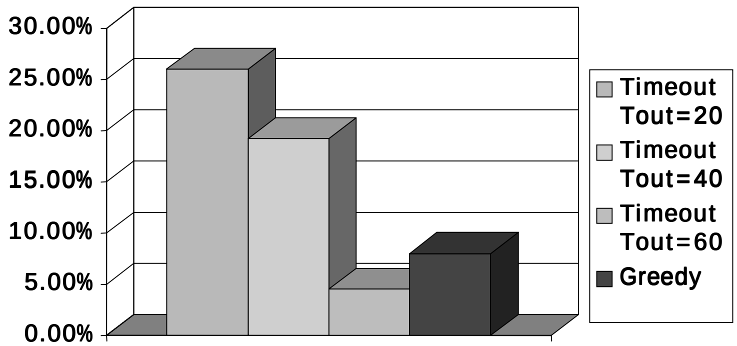
- System model
  - A SP with three power mode
  - A SR model with two states r_1 and r_2, $G_{SR}(r_1,r_2)$=1/200, $G_{SR}(r_2,r_1)$=1/400, $\lambda_l(r_1)$=1/30, $\lambda_h(r_1)$=1/50, $\lambda_l(r_2)$=1/60, $\lambda_h(r_2)$=1/90
  - A SQ model with a LSQ of length 3 and a HSQ of length 2
- Two different workload trace
  - Exactly same as theoretical model (exponential distribution)
  - Uniform distribution of request inter-arrival time (instead of exponential distribution)

Results for Trace 1



Power Improvement of Our approach vs. Heuristics

Results for Trace 2

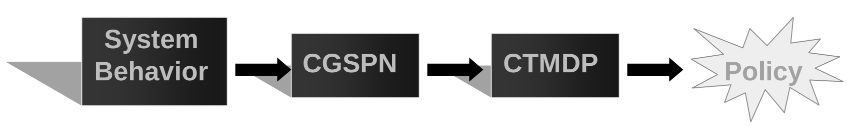


Power Improvement of Our approach vs. Heuristics

- Nearly same delay values

Generalized Stochastic Petri Nets For Complex DPM System

- CTMDP is not efficient in modeling complex systems
  - Need to construct system model manually
- Generalized Stochastic Petri Nets (GSPN)
  - Graphical tool for the formal description of complex system
  - Widely used in complex system performance analysis
  - Construction is straightforward from system behavior
  - Captures synchronization, mutual exclusion and conflict information easily
  - GSPN can be transformed to CTMP
- Controllable Generalized Stochastic Petri Nets (CGSPN)
  - CGSPN can be transformed to CTMDP

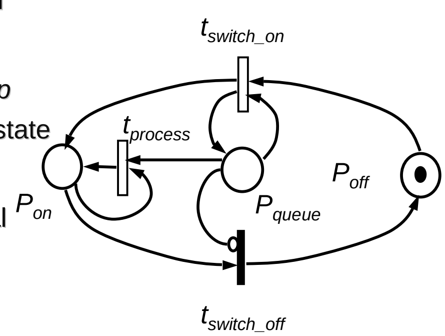


Outline of Part III

- GSPN background
- Finding the embedded CTMP of a GSPN
- Introducing Controllable GSPN
- Complex power managed system modeling
  - Component modeling
  - Entire system modeling

GSPN Primitives

- Place: condition or situation
- Token
  - Marking ***m***(p): #of tokens in p
  - System marking ***m***: system state
- Transition: events
  - Timed transition (exponential distribution) $R(t)$
  - Immediate transition
- Input arc: I(t, p)
  - $t\in p^\bullet$, $p\in {}^\bullet t$
- Output arc: O(t, p)
  - $t\in {}^\bullet p$, $p\in t^\bullet$
- Inhibitor arc: H(t, p)
  - $t\in {}^\circ p$, $p\in {}^\circ t$

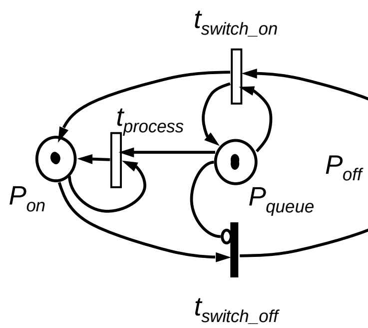


$\boldsymbol{m}(p_{on})=0$, $\boldsymbol{m}(p_{off})=0$, $\boldsymbol{m}(p_{queue})=0$

$\boldsymbol{m} = [0, 0, 1]$

GSPN Enabling and Firing Rules

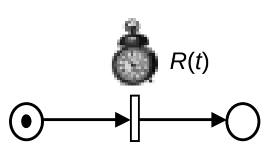
- *t* is enabled in marking ***m*** iff
  - $\forall p \in {}^{\bullet}t$, $\boldsymbol{m}(p) \geq I(t, p)$ and $\forall p \in {}^{\circ}t$, $\boldsymbol{m}(p) < H(t, p)$
- Firing of *t*
  - Removes $I(t, p)$ tokens from ${}^{\bullet}t$
  - Deposits $O(t, p)$ tokens into $t^{\bullet}$



p_{on}	p_{off}	p_{queue}
0	1	1
1	0	1
1	0	0
0	1	0

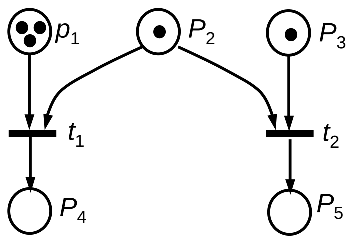
GSPN Enabling and Firing Rules (cont.)

- A timer is associated with timed transition *t*
  - When t is enabled, timer is set to a random value and starts counting down
  - When timer reaches 0, *t* fires and resets the timer
- Immediate transition always has higher priority than timed transition
  - tangible marking: no immediate transition is enabled
  - vanishing marking: at least one immediate transition is enabled

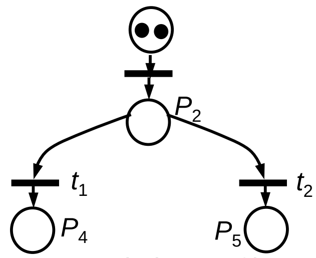


Conflict Transitions

- Effective conflict
  - Firing of one transition will disable another enabled transition
- Free choice conflict
  - Effective conflict transitions, which are always enabled at the same time



Effective conflict



Free choice conflict

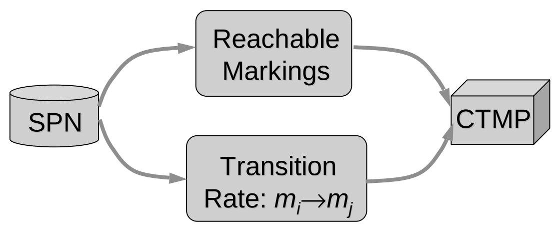
Resolving Conflict

- Conflict timed transitions
  - Transition with the shortest associated time fires first
- Conflict immediate transitions
  - Transition fires under randomized choice
  - Each conflict immediate transition is associated with a weight w_i
  - The probability of firing an immediate transition t_k is

$$P(t_k \mid \boldsymbol{m}) = \frac{w_k}{\sum_{t_j \text{ is enabled in } \boldsymbol{m}} w_j}$$

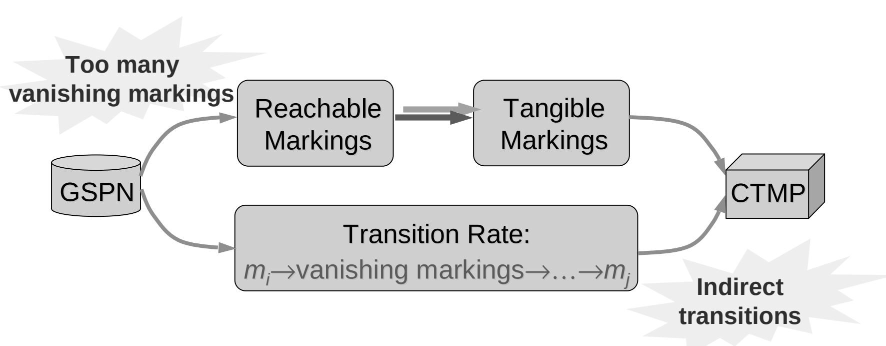
SPN and CTMP

- SPN is GSNP without immediate transitions
- SPN with a finite reachability set is isomorphic to a CTMP
  - CTMP states space: the reachable markings of SPN
  - σ_{m_i,m_j} : sum of the rate of transitions which moves SPN from m_i to m_j



GSPN and CTMP

- For each GSPN with finite reachability set, there is a unique embedded CTMP
  - State space: tangible markings of GSPN
  - Steady state probability and state transition probability are the same as that of the tangible markings in the GSPN



Convert GSPN to SPN

- Eliminate all of the vanishing markings by removing immediate transitions and vanishing places from the PN model

GSPN With Cost

- Impulse cost:
  - Associated with transitions
- Rate cost
  - Associated with places
- Can be converted to CTMP with cost
  - Rate cost:

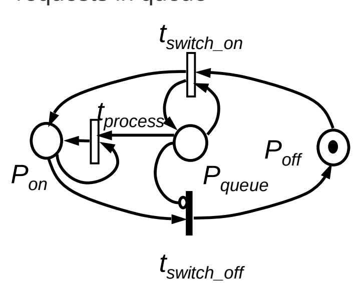
$$r_{\mathbf{m}} = \sum_{p \in P} c(p)$$

  - Transition cost:

$$r_{\mathbf{m},\mathbf{m}'} = \sum_{t:\mathbf{m}[t>\mathbf{m}'} c(t)$$

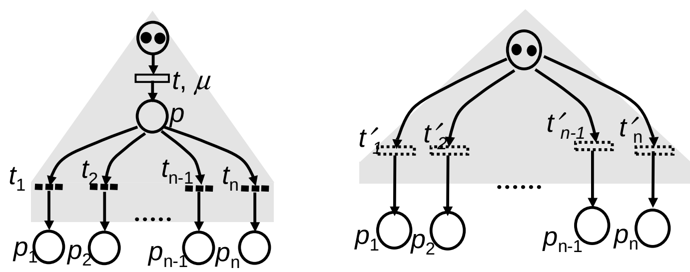
$c(t_{switch\_on})$=2J: Energy for turn on
$c(t_{switch\_off})$=0.1J: Energy for turn off

$c(p_{on})=\mathbf{m}(P_{on})\cdot$2.5W: "on" power
$c(p_{off})=\mathbf{m}(p_{off})\cdot$0.1W: "off" power
$c(p_{queue})=\mathbf{m}(p_{queue})$: #of waiting requests in queue



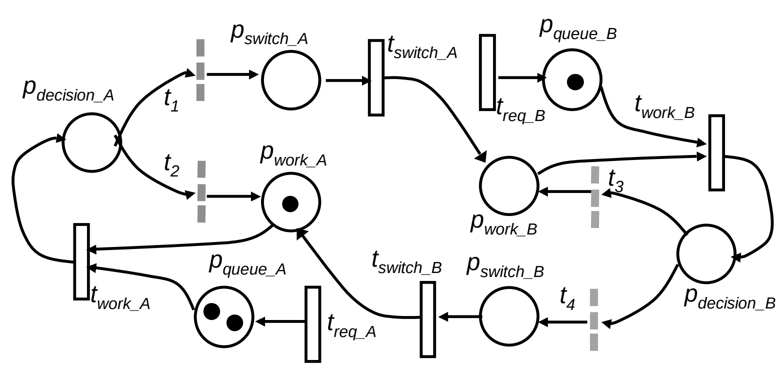
Controllable GSPN

- A controllable GSPN is a GSPN where the weights of all or part of free-choice-conflict immediate transitions can be controlled by outside commands
  - Corresponds to a controllable CTMP
  - Need to find the set of weights that minimizes the cost



$$\sigma^{a_i}_{\mathbf{m},\mathbf{m}'} = R(t) \quad (t\text{: } \mathbf{m} \rightarrow \mathbf{m})$$

Example of Controllable GSPN

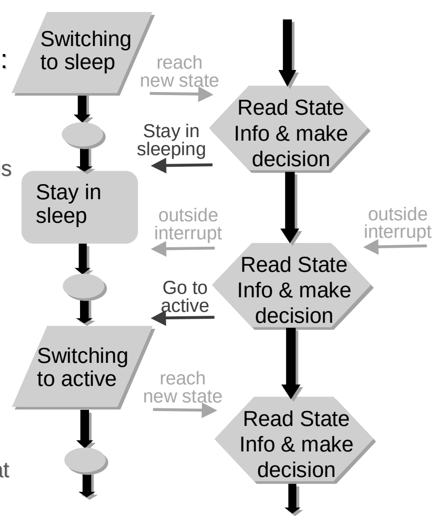


Unit Server System Modeling: Basic Elements

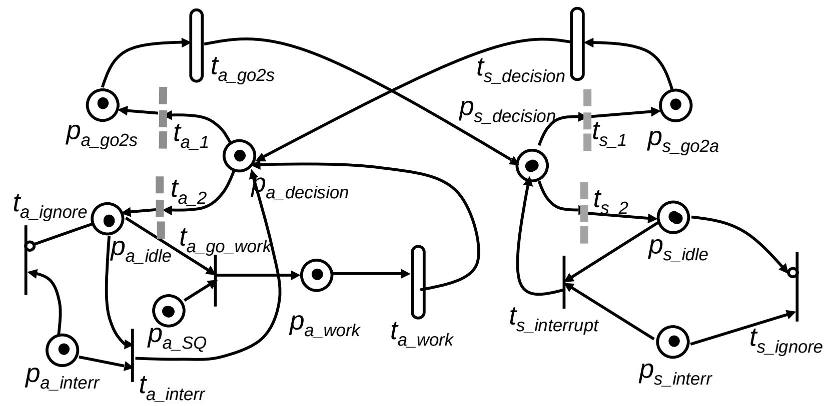
- The CGSPN model contains
  - Places set: $\{p_{[power]\_[serv]\_[status]}\}$
    - Models SP status
    - One for each different power mode, different service speed or different SP state (busy, idle or switching)
  - Places p_{SQ}
    - Models service queue
  - Timed transitions: $\{t_{[power]\_[serv]\_work}\}$
    - Models service providing procedure
  - Immediate transitions $\{t_{[power]\_[serv]\_go\_work}\}$
    - Synchronizes SP and SQ
  - Timed transitions $\{t_{[SP]\_[power]\_[serv]\_[switch]}\}$
    - Models the activity that the SP switches from one power mode to another

Unit Server System Modeling: Power Management Elements

- To model power management procedure, the GSPN must contains:
  - $p_{[power]\_[serv]\_decision}$,
    - Vanishing place
    - Models the short period when SP is receiving command
  - $p_{[power]\_[serv]\_interrupt}$
    - Vanishing place
    - $\boldsymbol{m}(p_{interrupt})$=1, there is interrupt
    - M($p_{interrupt}$)=0, no interrupt
    - $^{\bullet}p_{interrupt}$: sensitivity events
  - $t_{[power]\_[serv]\_[status]}$
    - Controllable immediate transitions
    - Models the switching command that will be issued by PM



Example of the Unit Server System



p_mode	status
active	Idle, go2s
sleep	Idle, go2a

Power mode: active (a), sleep (s)
Single service type
Controllable transitions: $\{t_{a\_1}, t_{a\_2}\}$, $\{t_{s\_1}, t_{s\_2}\}$
PM recheck the system state every time reaching a new state

Request Generating System

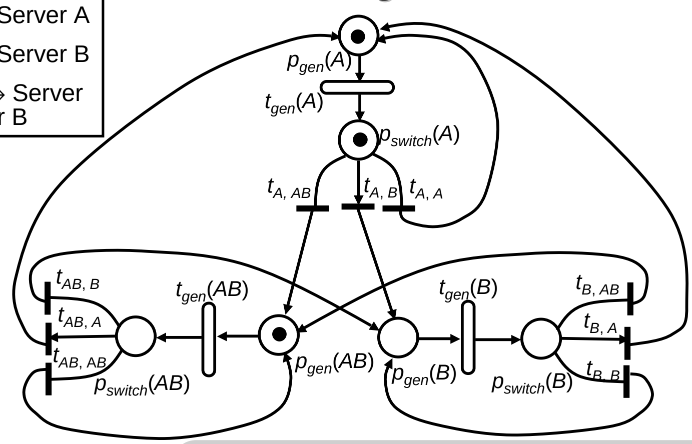
- The RGS generates different types of requests
  - Request can be serviced by one or more Service Provider
- The request generation takes some amount of time
  - Request inter-arrival time
- The temporal correlation between different types of requests can be modeled
  - prob(*j*| *i*) is known
- The request generation is stopped if SQ is full

Example of Request Generating System Modeling

Type A → Server A

Type B → Server B

Type AB → Server A or Server B



Two SP's: A, B
SQ_A capacity = 5
SQ_B capacity = 3

$$D_{t_{gen}(A)} = F_1(p_{gen}(A),1) \wedge F_2(p_{A\_SQ},5)$$

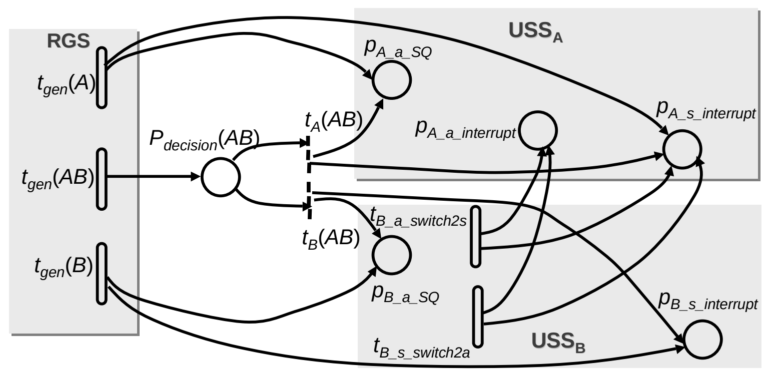
$$D_{t_{gen}(B)} = F_1(p_{gen}(B),1) \wedge F_2(p_{B\_SQ},3)$$

$$D_{t_{gen}(AB)} = F_1(p_{gen}(AB),1) \wedge (F_2(p_{B\_SQ},3) \vee F_2(p_{A\_SQ},5))$$

Entire System

- Connect unit server and requestor generator models with input/output/inhibitor arcs
  - If type *i* request can only be serviced by one SP: "a" in power mode "*p*" with service type "*serv*"
    - Connect $t_{gen}(i)$ to $p_{a\_p\_serv\_SQ}$
  - If type *i* request can be serviced by multiple SP's
    - A place $p_{decision}(i)$
    - A set of immediate transitions $t_{SP}(i)$
    - $t_{SP}(i)$ may be controllable to model the request dispatcher
    - $^\bullet t_j(i) = p_{decision}(i)$, $t_j(i)^\bullet = p_{[SP]\_[power]\_[serv]\_SQ}$, $^\bullet p_{decision}(i) = t_{gen}(i)$
  - Connect sensitivity transitions to $p_{[sp]\_[power]\_[serv]\_interrupt}$
  - Captures the interaction between the server and the request generator

Example of Muti-server System Modeling



- SPA is sensitive to the power mode of SPB
- SPB is not sensitive to the power mode of SPA
- Both are sensitive to the incoming of request

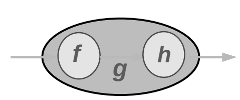
Non-exponential Distribution

- The GSPN model requires that each timed transition follows an exponential distribution
- Approximate the non-exponential distribution using the stage method

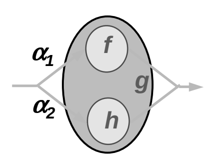
$g(t) \xrightarrow{L(s)} G(s)$

$f(t) \xrightarrow{L(s)} F(s)$

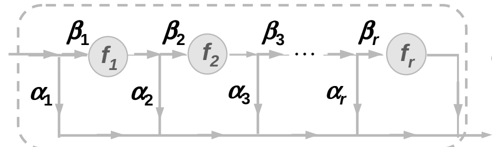
$h(t) \xrightarrow{L(s)} H(s)$



$G(s)=F(s) \cdot H(s)$



$G(s)=F(s) + H(s)$

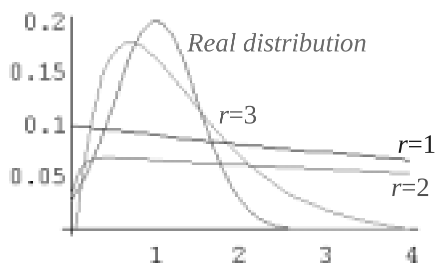


$$G(s) = \alpha_1 + \sum_{i=1}^{r} \beta_1 \beta_2 \cdots \beta_i \alpha_{i+1} \prod_{j=1}^{i} \left(\frac{\mu_j}{s+\mu_j} \right)$$

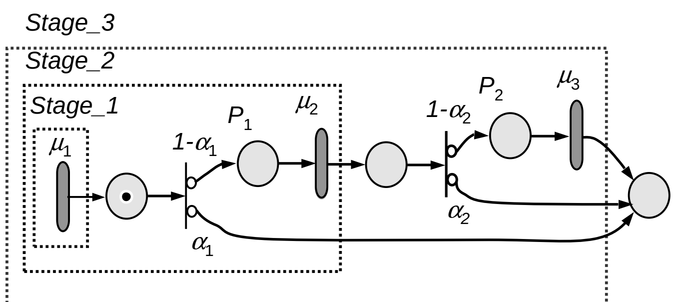
$\alpha_i + \beta_i = 1,\ 1 \le i \le r$

Real Implementation of Stage Method

- In real implementation, r = 3
- Use curve fitting to determine α_i and μ_i, $1 \le i \le r$



GSPN Model for Non-exponential timed activity

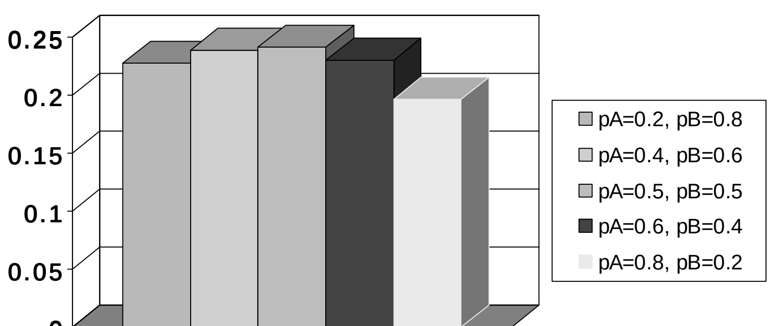


- We use the stage method to approximate the non-exponential inter-arrival time of requests with r = 3

Experimental Results: Complex DPM System

- System model
  - Two SP's (SP_A and SP_B) has the same functionality
  - SPA
    - Average service time: 5ms
    - pactive=2.3w, $p_{waiting}$=0.8w, $p_{sleeping}$=0.1w
  - SP_B
    - Average service time: 3ms
    - p_{active}=4.0w, $p_{waiting}$=0.8w, $p_{sleeping}$=0.1w
  - Two SQ's each with capacity two
  - Request can be serviced by both SP's
  - The switching time and energy for both SP's are also known

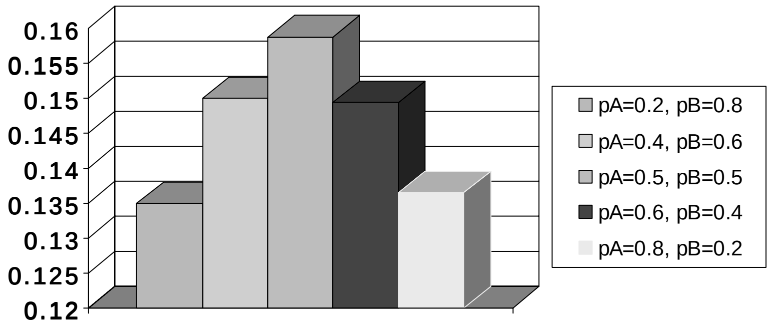
Comparison Results (I)



Power Improvement of Our approach vs. Heuristics

- Base case:
  - Greedy DPM policy for the servers
  - Randomized policy (pA, pB) for the dispatcher
- More than 20% power saving

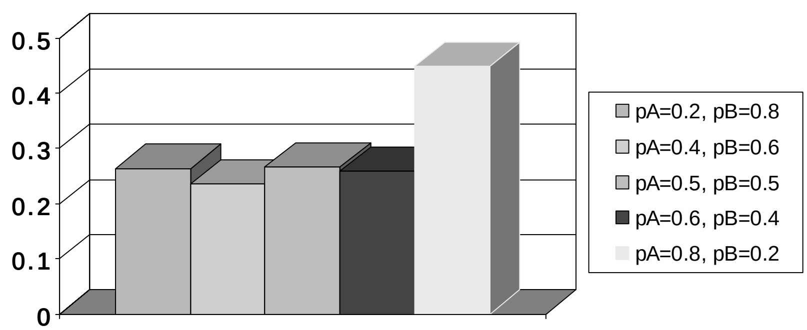
Comparison Results (II)



Power Improvement of Our approach vs. Heuristics

- Base case:
  - Timeout DPM policy for the servers
  - Randomized policy (pA, pB) for the dispatcher
- More than 13% power saving

Comparison Results (III)



Power Improvement of Our approach vs. Heuristics

- Base case:
  - Local optimal DPM for the servers
  - Randomized policy (pA, pB) for the dispatcher
- More than 20% power saving

Conclusions

- We introduced a new and complete model for simple and complex power managed systems
- Policy optimization techniques based on the proposed system model were presented
- The proposed dynamic power management methods outperform the existing approaches