System-Level Power Management

Tajana Simunic
Stanford University & HP Labs
Energy-efficient computing is needed by:
- portable systems
  - increase battery lifetime
  - optimize thermal design -> form factor
- non-portable systems
  - minimize cost
  - environmental concerns

Electronic system design
- hardware: processing, storage, communication
- software: operating systems, applications

Electronic system utilization
- run-time control and management
- e.g. control of device power and performance states
Systems and components are:
- Designed to deliver peak performance, but …
- Not needing peak performance most of the time

Dynamic power management (DPM):
- Shut-down idle components

Dynamic voltage scaling (DVS)
- Slow-down components - scale frequency and voltage
Portable computers

- Most energy consumed in: display, hard disks, WLAN

**Hard Disk**
- Active power: 0.95 - 2.5 W
- Idle power: 0.95 W
- Sleep: 0.13 W
- Sleep time: 0.67 s
- Wake-up time: 1.6 s

**WLAN**
- Transmission: 1.65 W
- Receiving: 1.4 W
- Doze: 0.045 W
- Down time: 62 ms
- Wake-up time: 34 ms
SmartBadge (HP)

Component-based wearable system

- Active power: 3.5 W
- Idle power: 2.2 W
- Standby: 0.2 W
- Sleep time: 1ms
- Wake-up time: 150 ms

Microphone and Speakers

Memory: Flash (1MB) SRAM (1MB)

StrongARM SA-1100

PCB

UCB1200

Analog & Digital Sensors

Display

RF

DC-DC Converter

Battery

Active power: 3.5 W
Idle power: 2.2 W
Standby: 0.2 W
Sleep time: 1ms
Wake-up time: 150 ms
Power manageable components

- Components with several internal states
  - Corresponding to power and service levels
- Abstracted as **power state machines**
  - State diagram with:
    - Power and service annotation on states
    - Power and delay annotation on edges

**Example: SA-1100**

- **RUN**: operational
- **IDLE**: a sw routine may stop the CPU when not in use, while monitoring interrupts
- **SLEEP**: Shutdown of on-chip activity
Example: Hard disk drive

Fujitsu MHF 2043 AT

Working: 2.2 W (spinning + IO)

Idle: 0.95 W (spinning)

Sleeping: 0.13 W (stop spinning)

read / write

IO complete

spin up
4.4J, 1.6 sec

shut down
0.36 J, 0.67 sec

Tajana Simunic
Structure of power-manageable systems

- System consists of several components:
  - E.g., Laptop: processor, memory, disk, display …
  - E.g., SOC: CPU, DSP, FPU, RF unit

- Components may:
  - Self-manage state transitions
  - Be controlled externally

- Power manager (PM):
  - Abstraction of power control unit
  - Implemented typically in software
  - Energy consumption of PM is negligible
The applicability of DPM

- State transition power ($P_{tr}$) and delay ($T_{tr}$)

- If $T_{tr} = 0$, $P_{tr} = 0$ the policy is trivial
  - Stop a component when it is not needed

- If $T_{tr} \neq 0$ or $P_{tr} \neq 0$ (always...)
  - Shutdown only when idleness is long enough to amortize the cost
  - What if $T$ and $P$ fluctuate?
System break-even-time: $T_{BE}$

Minimum idle time for amortizing the cost of component shutdown

$$T_{BE} = T_{tr} + T_{tr} \frac{P_{tr} - P_{on}}{P_{on} - P_{off}}$$

Transition delay ($T_{tr}$)  
Transition power ($P_{tr}$)  
Sleep power ($P_{off}$)
Effect of $T_{BE}$ & $F(T_{idle})$ on power savings

$$E_{saved} = (P_{on} - P_{off}) \cdot (T_{avg}^{idle > T_{BE}} - T_{BE}) \cdot (1 - F(T_{BE}))$$

- Power reduction in sleep state
- Expected useful idle time
- Probability of going to sleep

$E_{saved}$: Power reduction in sleep state

$T_{BE}$: Time to boot

$T_{avg}^{idle > T_{BE}}$: Average useful idle time

$P_{on}$: Power on

$P_{off}$: Power off

$F(T_{BE})$: Probability of going to sleep
When to use power management

- When $T_{BE} < T_{avg \ idle}$
  - Average idle periods are long enough
  - Transition delay is short enough
  - Transition power is low enough
  - Sleep power is low enough

- When designing system for a known workload
  - Criteria for component specification and design
Controlling PM systems

- DPM is a control problem: a policy is the control law
  - Collect observations
  - Issue commands
- Optimal control
  - Synthesize the "best" controller (PM)
Measurements done on Fujitsu MHF 2043 AT hard disk
Predictive techniques

- Observe time-varying workload
  - Predict idle period $T_{\text{pred}} \sim T_{\text{idle}}$
  - Go to sleep if $T_{\text{pred}}$ is long enough to amortize state transition cost

- Use predictive techniques when workload has memory

- Implementing predictive schemes
  - Predictor families must be chosen based on workload types, with parameters tuned to the instance-specific workload statistics
  - When workload is non-stationary, on-line adaptation is required

- Main issue: prediction accuracy
Fixed time-out

- Simple policy
  - If $T_{idle} > T_{TO}$ go to SLEEP
  - Stay in sleep until workload $\neq 0$

- Rationale
  - When $T_{idle} > T_{TO}$ it is likely that: $T_{idle} > T_{TO} + T_{BE}$

- Choice of $T_{TO}$ is critical
  - Large is safe, but it could be useless
  - Too small is highly undesirable

- Limitations
  - Performance penalty for wake-up is paid after every shutdown
  - Power is wasted during $T_{TO}$
Component-specific time-out [Karlin et al.]

- Choose: \( T_{TO} = T_{BE} \)
- Property:
  - Worst-case energy consumption is twice higher as compared to an oracle policy which knows the future
- Rationale:
  - Worst case workload has repeated idle periods of length \( 2T_{BE} \) separated by point-wise activity
- Disadvantages:
  - For some components (e.g., hard disks) a small time-out corresponds to many transitions, which affect component reliability
  - Time-out policies do not consider latency penalty
Predictive shutdown
(Srivastava & Broderson)

- More aggressive than time-out
  - Eliminates power waste caused by $T_{TO}$
  - Shutdown as soon as idle, if predicted idle period $T_{pred} > T_{BE}$
- Prediction is based on past history
  - Non-idle periods $T_{active}$ are observed as well
  - $T_{idle}^n, T_{active}^n$: n-th idle and active periods
  - Short active periods are often followed by long idle periods

If $T_{active}^{n-1} < T_{a-threshold}$ then go to SLEEP
Stochastic control

- Recognize inherent uncertainty
  - Exact prediction of future events is impossible
  - Abstraction of system model implies uncertainty
- Model system and workload as stochastic processes
- Formulate system model based on Markov processes
- Optimize expected values of cost metrics (e.g. energy)
- Optimal control is non-deterministic if constraints are used (e.g. performance)
Stochastic System Model

Service Requestor  Queue  Service Provider

Active  Idle  Active  Idle  Sleep

Power Manager  Policy

System

4 3 2 1
Controlled Markov processes

- Component and workload modeled as Markov chains
  - Component is called service provider (SP)
  - Workload is called service requestor (SR)
  - System (S) is the combination of SR and SP (with queue)
- SP is a controlled Markov chain:
  - State transition probabilities depend on commands
- The power manager (PM) observes the state of the system and issues commands to control evolution

```
SR ——— PM ——— SP (with queue)
```
Discrete-time, finite-state CMPs

- Discrete time $t = 1, 2, \ldots$
  - System evaluated at periodic time points
- SR and SP are modeled by Markov chains
- PM can issue a finite number of commands $a$ in $A$

SR and SP are modeled by Markov chains.

$\begin{align*}
\text{SP} & \quad \{a=\text{GoON},\text{SR}=0\}: 0.5 \\
& \quad \{a=\text{GoSLEEP},\text{SR}=0\}: 0.0 \\
& \quad \{a=\text{GoON},\text{SR}=1\}: 0.0 \\
& \quad \{a=\text{GoSLEEP},\text{SR}=1\}: 0.0
\end{align*}$

$0.7$ $0.3$ $0.8$ $0.2$

State 0 = no request
State 1 = request

$A = \{\text{GoON, GoSLEEP}\}$

State O0 = On, no req. waiting
State O1 = On, 1 req. waiting
State S1 = Sleep, 1 req. waiting
State S0 = Sleep, 0 req. waiting
Power management policies

- PM observes system state and issues a command
- A policy is a sequence of commands
- A Markovian policy yields commands as function of system state (and not previous history)
- A deterministic policy
  - For each state $s$ in $S$, policy specifies command $a$ in $A$
- A randomized policy
  - For each state $s$ in $S$, command $a$ in $A$, policy specifies the probability of issuing $a$ in state $s$
- A stationary policy
  - The policy does not change with time
PM policy optimization

- Solve a stochastic optimal control problem:
  - Find a policy that
    - Minimizes power cost function
    - Satisfies performance constraints

- Dual formulation

- Key result for CMPs:
  - Optimum policy is stationary, Markovian and randomized
  - Policy optimization can be reduced to a LP and solved exactly and efficiently

![Diagram with axes labeled Power and Tolerable delay, showing Pareto curve, feasible solutions, and validation points.]
Controlled Markov processes advantages

- **Constrained optimization:**
  - Energy/performance (latency) trade-off

- **Global view of the system:**
  - Workload and component models

- **Optimum policy is captured by commands:**
  - Control policy is a table
  - Policy implementation is easy

- Policy computation can be cast as linear program and solved exactly and efficiently

- Easily expands to handle dynamic voltage scaling
Controlled Markov processes limitations

- Discrete-time models require periodic evaluation
  - Use continuous-time Markov models
  - Event-driven paradigm

- Stochastic distributions:
  - Geometric and exponential distribution of events may not fit component and workload
  - Use time-indexed semi-Markov models

- Non-stationary workloads
  - Use adaptive schemes
Event-driven stochastic models

- Continuous time models
  - All stochastic process have to be exponential

- Semi-Markov models:
  - One stochastic process may not be exponential (e.g. device transition times)

- Time-indexed semi-Markov models:
  - Support multiple non-exponential processes using time indexing
  - More complex to handle due to larger state space

- Renewal time model
  - Support multiple non-exponential process
  - Limitation: one decision point
Service Provider - Transition

CDF probability

Experimental
Uniform
Exponential

Transition time (ms)

1130 1330 1530 1730 1930 2130

$E_{SP} = \begin{cases} 
\frac{t-t_{\text{min}}}{t_{\text{max}}-t_{\text{min}}} & \text{if } t_{\text{min}} < t < t_{\text{max}} \\
0 & \text{else}
\end{cases}$

Taiana Simunic
Average service time = 0.008 s
Average fitting error = 6.8%

\[ E_{SP} = 1 - e^{-\lambda_{SP} t} \]
Service Requestor - Idle State

Pareto Distribution:
$$E_{SR} = 1 - a \cdot t^{-b}$$
Average interarrival time = 0.723 s
Average fitting error = 13%

\[ E_{SR} = 1 - e^{-\lambda_{SR}t} \]
average queue size = 0.8
average error = 3%

Queue state = number of jobs currently in the queue

System in idle state on empty queue
Renewal Model

- allows for only one decision state ("renewal point")
- policy optimization is formulated using "renewal time" concept
- optimizing energy under performance constraint or vice versa
- guarantees optimal results (linear program)
Define:

- $T$: expected renewal time
- $T_a$: time of first request arrival
- $j h$: time of transition to sleep from idle state
  (j=index, h=time increment)

$$E[T] = E[T | T_a < j h] + E[T | T_a > j h]$$

- $E[\text{Length of idle period}]$ + $E[\text{Time service request}]$
- $E[\text{Length of idle period}] + E[\text{Time to sleep}]$ + $E[\text{Length of sleep}]$ + $E[\text{Time to active}]$ + $E[\text{Time to service all requests}]$
calculated for each state using a **constant** $C_i$ (e.g. power consumption in state $i$), **expected time spent in the state**, and **probability of first request arrival**

<table>
<thead>
<tr>
<th>State</th>
<th>$C_i$</th>
<th>$E[T_{idle}]$</th>
<th>$P(T_a &lt; jh)$</th>
<th>$P(T_a \geq jh)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>$C_{iIdle}$</td>
<td>$E[T_{idle}]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sleep</td>
<td>$C_{iSleep}$</td>
<td>$E[T_{Sleep}]$</td>
<td>$P(T_a \geq jh)$</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>$C_{iActive}$</td>
<td>$E[T_{Active}]$</td>
<td>$P(T_a \geq jh)$</td>
<td>$P(T_a &lt; jh)$ + $P(T_a \geq jh)$</td>
</tr>
<tr>
<td>SrvIdle</td>
<td>$C_{iSrvIdle}$</td>
<td>$E[T_{SrvIdle}]$</td>
<td>$P(T_a \geq jh)$</td>
<td></td>
</tr>
<tr>
<td>SrvSleep</td>
<td>$C_{iSrvSleep}$</td>
<td>$E[T_{SrvSleep}]$</td>
<td>$P(T_a \geq jh)$</td>
<td></td>
</tr>
<tr>
<td>SrvToSleep</td>
<td>$C_{iSrvToSleep}$</td>
<td>$E[T_{SrvToSleep}]$</td>
<td>$P(T_a \geq jh)$</td>
<td></td>
</tr>
<tr>
<td>SrvToActive</td>
<td>$C_{iSrvToActive}$</td>
<td>$E[T_{SrvToActive}]$</td>
<td>$P(T_a \geq jh)$</td>
<td></td>
</tr>
</tbody>
</table>

$A, Q$
Basic assumptions:
- general distribution governs the first request arrival
- exponential distribution represents arrivals after the first arrival
- user, device and queue are stationary

Optimize average performance under average power constraint
- randomized policy

\[
\begin{align*}
\min & \quad \sum_j \text{Perf}(j)p(j) \\
\text{s.t.} & \quad \sum_j (\text{Energy}(j) - \text{Power}(j)T(j))p(j) = 0 \\
& \quad \sum_j p(j) = 1
\end{align*}
\]

Obtain globally optimal policy using linear programming
TISMDP Model
(Time-Indexed Semi-Markov Decision Process Model)

- allows multiple decision states (e.g., multiple low-power states)
- more general and more complex method
- guarantees optimal results
- base model is Semi-Markov decision process model
  - applies to states with at least one exponential transition
- time-indexing is needed to account for time in states where more than one non-exponential transition occurs
- same basic assumptions as with Renewal model:
  - general distribution governs the first request arrival
  - exponential distribution represents arrivals after the first arrival
  - user, device, and queue are stationary
Policy Optimization

- Optimize average power consumed under performance constraint
  
  - randomized policy
    
    $$\min \sum \sum \cos t_{\text{energy}} (s,a)f(s,a)$$
    
    $$\text{s.t. } \sum f(s,a) - \sum \sum m(s',s,a)f(s',a) = 0; \forall s'$$
    
    $$\sum \sum T(s,a)f(s,a) = 1$$
    
    $$\sum \sum \cos t_{\text{perf}} (s,a)f(s,a) < \text{Constraint}$$

- Obtain globally optimal policy using linear programming

- Policy format:
  
  - table of probabilities of issuing command a when system in state s
  
  - obtained from state-action frequencies, f(s,a):
    
    $$p(s,a) = \frac{f(s,a)}{\sum_{a' \neq a} f(s,a')}$$
Optimal Policy Implementation

- Power management implementation algorithm:
  - on entry to idle state:
    - obtain a random number RND
    - find first time $j_h$ for which $RND > p(j_h)$ holds
    - if no arrival during $j_h$ seconds
      - enter low-power state
    - else
      - enter active state
  - during transition
    - store user requests in the buffer
  - once in low-power state
    - wait until user request comes
    - place device in the active state

<table>
<thead>
<tr>
<th>Idle time (ms)</th>
<th>Probability to lp state</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j_h$</td>
<td>$p(j_h)$</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>0.12</td>
</tr>
<tr>
<td>30</td>
<td>0.43</td>
</tr>
<tr>
<td>40</td>
<td>0.75</td>
</tr>
<tr>
<td>50</td>
<td>0.87</td>
</tr>
<tr>
<td>60</td>
<td>0.91</td>
</tr>
<tr>
<td>70</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Dynamic Voltage Scaling

- Adjusts performance and energy consumption while the device is active
- Most systems are designed for continuous peak performance
- Key is to meet users performance needs while saving energy
- Reduce processor frequency and voltage to obtain quadratic energy savings

Example:
- SA-1100 CPU has 11 frequency and voltage settings
- Each (f, V) setting represents one active state in the system model
- Easy expansion of stochastic models
DPM with DVS Model

**Active State**
- queue > 0
- t < h

**Idle State**
- queue = 0
- h < t < 2h

**Sleep State**
- queue = 0
- t > jh

**Dynamic Power Management Model**

**Dynamic Voltage Scaling with DPM with DVS Model**

**Active State**
- f_0, V_0
- t < h

**Idle State**
- q = 0
- U < t < h + U

**Sleep State**
- q > 0
- U < t < h + U

**No Arrival**
- Sleep
- Active
- Idle
DVS has two main portions:

1. detection of the change in request arrival or servicing rates
2. the adjustment of CPU frequency and voltage

Change point detection:

- off-line characterization to obtain a maximum likelihood ratio for a set of all possible arrival and servicing rates
- run-time detection of the change in rate using the ratio

\[
\ln(P_{\text{max}}) = (n_{\text{points}} - n_{\text{change}} + 1) \ln \frac{\lambda_{\text{new}}}{\lambda_{\text{old}}} - (\lambda_{\text{new}} - \lambda_{\text{old}}) \sum_{j=n_{\text{change}}}^{n_{\text{points}}} t_j
\]

Adjustment of CPU frequency and voltage:

- obtained using M/M/1 queuing theory results
- average expected frame delay in the queue is kept constant

\[
\text{Frame}_{\text{delay}} = \frac{\lambda_{\text{device}}}{\lambda_{\text{user}}(\lambda_{\text{user}} - \lambda_{\text{device}})}
\]
Change Point Detection Results

- Detects when the rate of incoming user’s request changes
- Within 10 frames of ideal detection, and much better than standard exponential average detection:

\[ \lambda_{\text{new}} = (1 - g)\lambda_{\text{old}} + g\lambda_{\text{cur}} \]
Handling non-stationary workload

- Adaptive control scheme
  - Off-line policy computation for different parameters
  - On-line parameter estimation
  - On-line policy interpolation

- Works well because policies change smoothly with parameters

\[ P = \begin{bmatrix} 0 & 1 \\ 0.9 & 0.1 \end{bmatrix} \]

\[ P = \begin{bmatrix} 0 & 1 \\ 0.9 & 0.1 \end{bmatrix} \]

\[ R_i(t) : \text{User Request Probability} \]
**Policy Table Construction**

### Policy Table

<table>
<thead>
<tr>
<th></th>
<th>R₀(0)</th>
<th>R₁(0)</th>
<th>R₁(1)</th>
<th>R₁(2)</th>
<th>R₁(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₀</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Decision Table

**S: System States**  
**A: Commands**

<table>
<thead>
<tr>
<th></th>
<th>A₀</th>
<th>A₁</th>
<th>A₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₀</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>S₁</td>
<td>0.9</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>S₂</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>
On-line parameter estimation

- Based on previous history
  - Predict $P$ (transition matrix of $SR$)
- Window-based approach

$W_i(j)$ : Window Slot

Window Size : Critical for precision

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...........</td>
</tr>
</tbody>
</table>

$W_i$
Interpolation procedure

Policy Table

\[
\begin{array}{cc}
R_1(2) & R_1(3) \\
R_0(3) & \ (PR_0, PR_1) \\
R_0(4) & \\
\end{array}
\]

\[
R_0(3) \leq PR_0 < R_0(4) \\
R_1(2) \leq PR_1 < R_1(3)
\]
In systems with an operating system (OS):
- The OS knows of tasks running and waiting
- The OS should perform the DPM decisions

Advanced Configuration and Power Interface (ACPI) [Intel, Microsoft, Toshiba]
- Open standard for design of OS-based power management

ACPI supported by:
- Recent PCs (1999+)
- Windows 2000

Most PC still use APM

Experiments with:
- Sony VAIO PCG-F150
- Power savings: 1.7 X
- Comparable performance
ACPI architecture

- Kernel
- Power Management
- Device Driver
- ACPI driver
- AML interpreter
- BIOS interface
- ACPI Tables
- ACPI BIOS
- ACPI registers
- Platform Hardware
- BIOS
- Motherboard devices
- Chipset
- CPU
DPM and operating systems

- **Device**
  - consumes power
  - should provide mechanism, not policy
- **Driver**
  - detects busy and idle periods
- **Process manager**
  - knows multiple requesters
  - can estimate idle periods more accurately
- **Scheduler**
  - selects processes and affects idle periods
- **Application**
  - should not directly control hardware power
  - does not support power management in legacy programs
Process-Based Policy Implementation

- for each process $p$
  - for each device $d$
    - estimate device utilization $U(d, p) = \frac{1}{T_{idle_n}}$
  - estimate CPU utilization $CPU(p) = \frac{CPUtiltime(p)}{\sum_{\forall p} CPUtiltime(p)}$
- for each device $d$
  - estimate device utilization $U(d)$ from all processes
    $$U(d) = \sum_{\forall p} U(d, p) \times CPU(p)$$

Shut down if $U(d) < \text{threshold} \; th$, let $th = \frac{k}{T_{be}}$

- $k = 1 \Rightarrow$ expected TBR is $T_{be}$
- $k < 1 \Rightarrow \; > T_{be}$ (conservative)
- $k > 1 \Rightarrow \; < T_{be}$ (aggressive)
Low-Power Scheduling

Tasks specify
- device requirements
- timing requirements

Example:
- autosaver,
- email download

Operating System:
1. Group tasks with same device requirements
2. Arrange groups with similar device requirements
3. Execute tasks in groups
4. Wake up devices in advance to meet timing constraints
Policy is implemented using ACPI standard on the Hard Disk of Sony Vaio laptop running Win NT 5.0β

measured real power consumption

11 hr user trace

within 11% of ideal oracle policy

factor of 2.4 lower than always-on

factor of 1.7 lower than default time-out

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Power (W)</th>
<th>Tss (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>0.33</td>
<td>118</td>
</tr>
<tr>
<td>TISMDP</td>
<td>0.40</td>
<td>81</td>
</tr>
<tr>
<td>Karlin's</td>
<td>0.44</td>
<td>79</td>
</tr>
<tr>
<td>30s Timeout</td>
<td>0.51</td>
<td>157</td>
</tr>
<tr>
<td>120s Timeout</td>
<td>0.67</td>
<td>255</td>
</tr>
<tr>
<td>Always on</td>
<td>0.95</td>
<td>0</td>
</tr>
</tbody>
</table>
## WLAN Results

Measurements with Lucent’s WLAN card on Linux laptop

### PM for WWW

<table>
<thead>
<tr>
<th>Policy</th>
<th>Nsd</th>
<th>Nwd</th>
<th>Tpenalty (sec)</th>
<th>Pave (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>395</td>
<td>0</td>
<td>0</td>
<td>0.467</td>
</tr>
<tr>
<td>Renewal</td>
<td>363</td>
<td>96</td>
<td>6.9</td>
<td>0.474</td>
</tr>
<tr>
<td>Karlin's</td>
<td>623</td>
<td>296</td>
<td>23.8</td>
<td>0.479</td>
</tr>
<tr>
<td>Poisson</td>
<td>3424</td>
<td>2866</td>
<td>253.7</td>
<td>0.539</td>
</tr>
<tr>
<td>Default</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.41</td>
</tr>
</tbody>
</table>

### PM for Telnet

<table>
<thead>
<tr>
<th>Policy</th>
<th>Nsd</th>
<th>Nwd</th>
<th>Tpenalty (sec)</th>
<th>Pave (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>766</td>
<td>0</td>
<td>0</td>
<td>0.22</td>
</tr>
<tr>
<td>Renewal</td>
<td>798</td>
<td>21</td>
<td>2.75</td>
<td>0.269</td>
</tr>
<tr>
<td>Karlin's</td>
<td>780</td>
<td>40</td>
<td>3.81</td>
<td>0.302</td>
</tr>
<tr>
<td>Poisson</td>
<td>943</td>
<td>233</td>
<td>20.53</td>
<td>0.361</td>
</tr>
<tr>
<td>Default</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.41</td>
</tr>
</tbody>
</table>

**PM savings of factor of 3**

**PM savings of factor of 5**

### Application BER

<table>
<thead>
<tr>
<th>Application</th>
<th>BER</th>
<th>( P_{ON}/P_{PMPC} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>1E-02</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td>1E-05</td>
<td>4.1</td>
</tr>
<tr>
<td>Telnet</td>
<td>1E-02</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>1E-05</td>
<td>6.7</td>
</tr>
</tbody>
</table>

**PM & PC savings of factor of 4 to 9**

---

Tajana Simunic
LP Scheduling & Process Based DPM

- No PM
- 3-min timeout
- Karlin
- Hwang
- Process-based
- Low-power scheduling

- Power (disk)
- Power (NIC)
- Delay (disk)
- Delay (NIC)
DPM & DVS Results

DVS is implemented on the SmartBadge with a WLAN card using MPEG2 video and MP3 audio streams for testing.

### MPEG Results

<table>
<thead>
<tr>
<th>Football (875s)</th>
<th>Energy</th>
<th>Ideal</th>
<th>Ch. Point</th>
<th>Exp. Ave.</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>214</td>
<td>218</td>
<td>300</td>
<td>426</td>
<td></td>
</tr>
<tr>
<td>Terminator2 (1200s)</td>
<td>Energy</td>
<td>280</td>
<td>294</td>
<td>385</td>
<td>570</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.11</td>
<td>0.16</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**MPEG2 with DVS**

- Factor of 2 energy savings

### MP3 Audio Results

<table>
<thead>
<tr>
<th>Clip 1</th>
<th>Energy</th>
<th>Ideal</th>
<th>Ch. Point</th>
<th>Exp. Ave.</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>196</td>
<td>217</td>
<td>225</td>
<td>316</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.09</td>
<td>0.1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Clip 2</td>
<td>Energy</td>
<td>189</td>
<td>199</td>
<td>231</td>
<td>316</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.09</td>
<td>0.1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Clip 3</td>
<td>Energy</td>
<td>190</td>
<td>214</td>
<td>232</td>
<td>316</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.04</td>
<td>0.1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**MP3 with DVS**

- Factor of 1.4 energy savings

### Algorithm Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Energy (kJ)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>4260</td>
<td>1</td>
</tr>
<tr>
<td>DVS</td>
<td>3142</td>
<td>1.4</td>
</tr>
<tr>
<td>DPM</td>
<td>2460</td>
<td>1.7</td>
</tr>
<tr>
<td>Both</td>
<td>1342</td>
<td>3.1</td>
</tr>
</tbody>
</table>

**DPM with DVS**

- Factor of 3.1 energy savings
Conclusions

- Power management can achieve large energy savings by exploiting variations in workload.

- Key issues:
  - Component and workload characterization
  - Policy computation

- Implementation choices:
  - Shutdown, clock gating, clock/voltage setting
  - Strong coupling between software, OS and hardware