Energy-Aware QoS Management

Faculty: Kang G. Shin
Grad students:
Padmanabhan Pillai
Hai Huang

Real-Time Computing Laboratory
EECS Department
The University of Michigan

Outline

- Real-time energy goals
- Energy-efficient services
- Real-time dynamic voltage scaling
- Memory power reduction
- Energy-aware Quality of Service (EQoS)

Motivation

- Increasing number of
  - handheld, mobile computation and communication devices
  - smart sensors, actuators and ammunitions
- Increasingly complex software and faster hardware, consuming more energy
- Rapid increases in HW complexity, speed, and power consumption, but battery technology is not keeping up
- Need to conserve energy, improve computational efficiency through the OS on power-constrained systems

Real-Time & Energy Goals

- Many power-constrained embedded or mobile systems have real-time tasks
  - Time/mission-critical computations, typically periodic
  - Need to provide guarantees for meeting deadlines
- Available stored energy fundamentally limits the system•s ability
- Need to allocate energy resource to most critical or desirable computations, while meeting timing constraints

Real-Time App Characteristics

- Typically, composed of well-defined task set
- Canonical model of a real-time task, T_i:
  - Is periodic, with period t_i
  - Has worst-case execution time, C_i
  - Has relative deadline, d_i typically equal to t_i
- Periodic model can accommodate aperiodic and sporadic tasks
- Schedulability of RT systems is well-studied.

Energy-Efficient RTOS: Accomplishments

- Reduce overhead in system services (SOSP•99)
  => lower computational overhead
  => lower CPU power consumption !!!
  - Optimized IPC for periodic RT tasks
  - Combined Static Dynamic (CSD) scheduling
  - Protocol stack layer-bypassing
  - Eliminate naming services
- Exploit HW mechanisms, e.g., voltage scaling of CPU (SOSP•01), power management of memory subsystem (USENIX•03)

RT-DVS

- Goal: reduce per-cycle CPU energy costs
- Reducing frequency permits lower voltage
- Lower voltage (V) on CPU to obtain V^2 savings per cycle
- Frequency change affects execution time, altering RT schedulability
- We have already developed energy-conserving algorithms for DVS that preserve RT guarantees (SOSP•01)

Memory Power Management

- Goal: reduce power dissipation for memory access
- Main memory consists of multiple devices, each with independently-controlled power states
- Switch devices not needed for current task to low-power states
- Modify page allocation to reduce the number of devices in use by each task
- 59-94% memory power reduction with RDRAM (USENIX•03)

Need for Adaptation

- Many existing techniques to reduce energy consumption
- No general guidelines on how to make best use of limited energy
- Want to provide more energy & runtime to more critical or beneficial tasks
- Need to adapt workload to maximize system gains or utility of computation

Example

- A remote surveillance device transmits compressed video and audio
- Solar-powered, but must run overnight
- 3 real-time tasks:
  - Radio transmitter (critical): constant bit rate
  - Video codec (degradable):
    high quality (30 fps, 640x480) MPEG4,
    low quality (10 fps, 160x120) MPEG1
  - Audio codec (noncritical): mp3, either on or off

Example, cont•d

- Adapt task set based on power consumption of tasks, available energy, hours until daylight, and relative value of the tasks, e.g.,
  - During daytime or high battery levels: radio, video at high quality, audio on
  - Low battery at night: radio, video low quality, audio on
  - Energy is critically low: radio, video low quality, audio off
- Dynamic adaptation needed in general, as battery levels and time until daylight are variable

EQoS

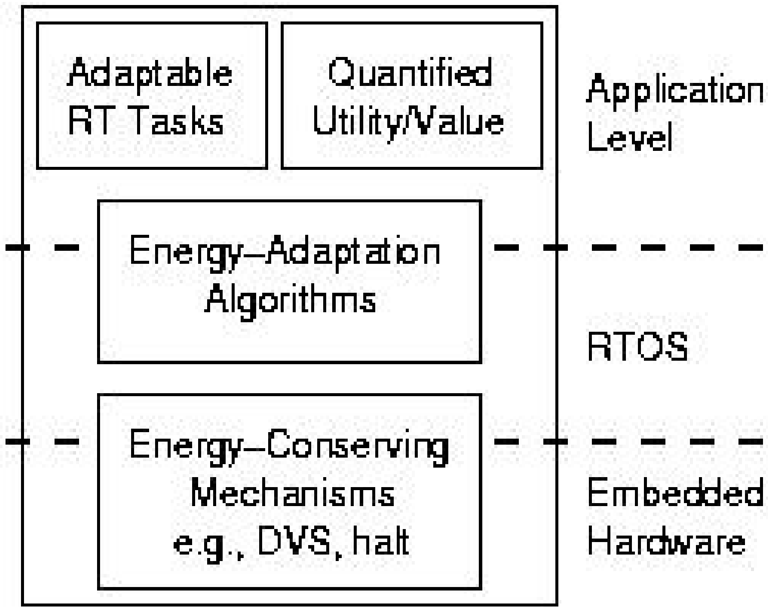
- Need to maximize benefits gained from energy spent, but HOW?

  => Energy-aware Quality of Service (EQoS):

  - Vary per-task QoS, which directly affects task energy consumption
  - Select a set of task QoS levels to maximize total utility of system over given runtime
  - Cast selection into tractable, maximization problem => MCKP

EQoS Design

- EQoS design goals:
  - Leverage sprint-and-halt and DVS techniques
  - Meet system runtime goals
  - Maximize benefits of task execution
- Need methods of changing QoS for RT tasks, and specifying benefits and energy requirements



RT QoS Adaptation

- How does one change QoS for RT tasks?
- Adapt techniques from RT & fault-tolerance:
  - Period extension
  - Imprecise computation
  - Apply different algorithms or CODECs
  - Omission
- Degraded service execution requires less energy
- For EQoS, need to specify set of QoS levels and required energy for each task

Utility

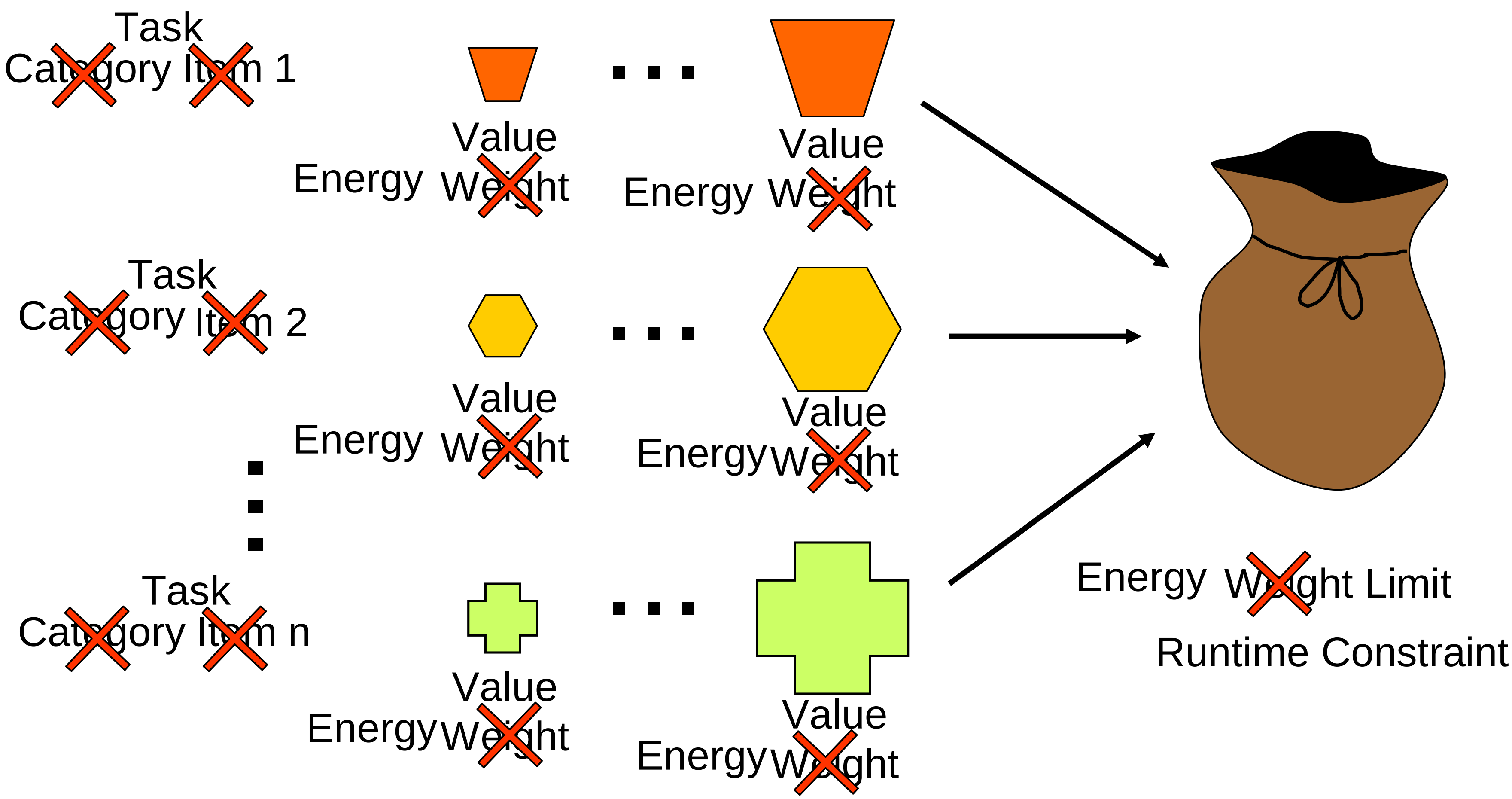
- Abstract notion of value from executing tasks
- Need to specify utility for each degraded service level of each task, e.g.,
  - Increasing Rewards for Increasing Service (IRIS)
  - Performance Index (PI) for control applications
  - Perceived-quality metrics for multimedia
- Actual specification flexible to types of applications and systems designed

EQoS Problem

- Given:
  - tasks with QoS levels defined, with energy required and utility gained for each level
  - remaining system energy
  - desired runtime, or known time until recharge
- Select a QoS level for each task, so as to:
  - achieve desired runtime
  - maximize total utility
- This can be formulated as a MCKP
  - Each task as a category and its set of QoS levels as items in the category
  - Knapsack size = power budget
  - Item values and weights = utility rates and power consumption

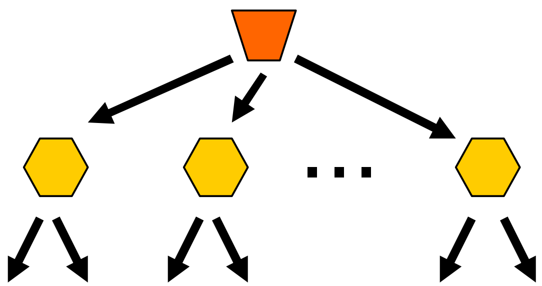
MCKP vs. EQoS Problem

EQoS ~~MCKnapsack~~ Problem



Optimal Algorithms

- NP-hard: all KP can be expressed as MCKP
- Exponential Search - $O(m^n)$
- Branch-and-Bound (BB)
  - Need fast bound computation
  - Can use LMCKP as upper bound
  - May still require exponential time

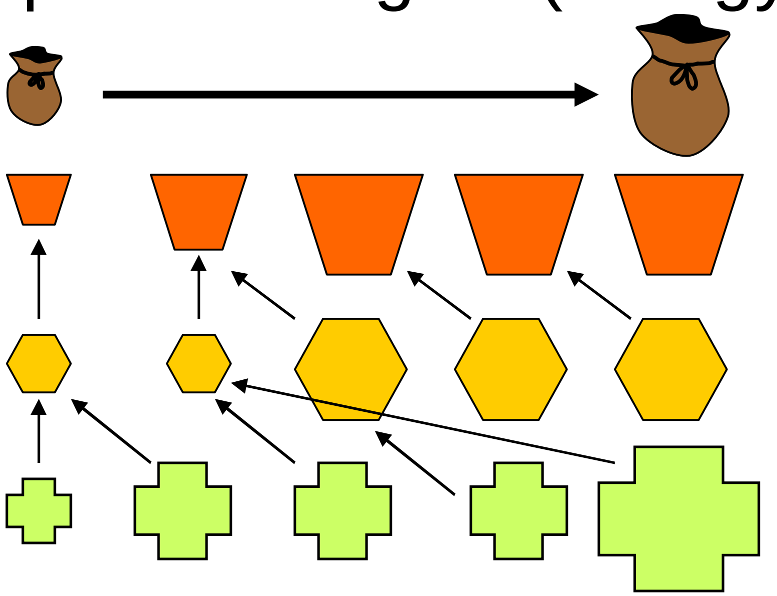


LMCKP Details

- Linear relaxation of MCKP - fractional selections allowed
  - Start with minimal QoS levels selected
  - Apply €upgrades• sorted by value/energy up to budget
  - Fractionally apply next upgrade
- Guaranteed ≥ discrete MCKP optimal solution
- O(nm) time, excluding sorting of upgrades

Optimal Algorithms, cont•d

- Dynamic Programming (DP)
  - Pseudo-polynomial time, O(mnk)
  - Partial solutions for 1, 2, , , n tasks for all possible power budgets (energy/runtime)



Heuristics

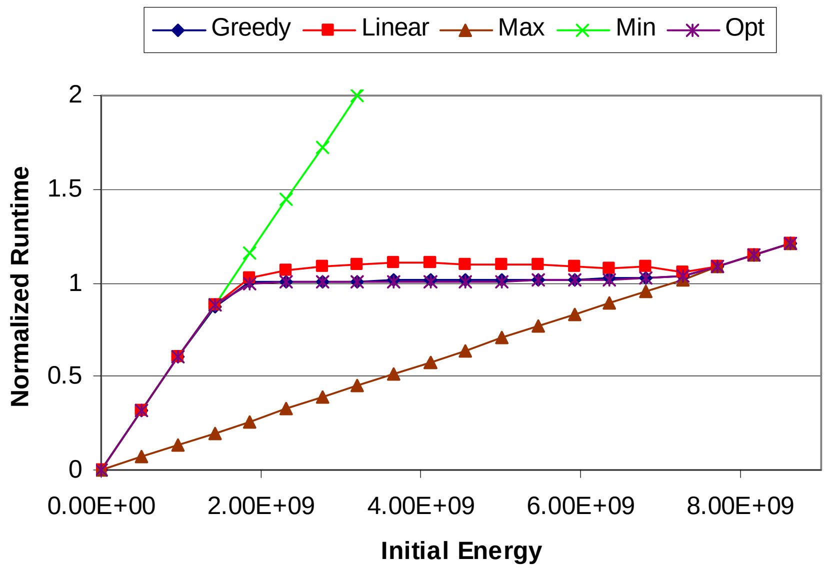
- Linear:
  - Use LMCKP solution, as with BB bound
  - Drop fractional part
- Greedy:
  - Start with same approach as LMCKP
  - Continue selecting smaller upgrades
- O(nm) overhead, without accounting for upgrades sorting

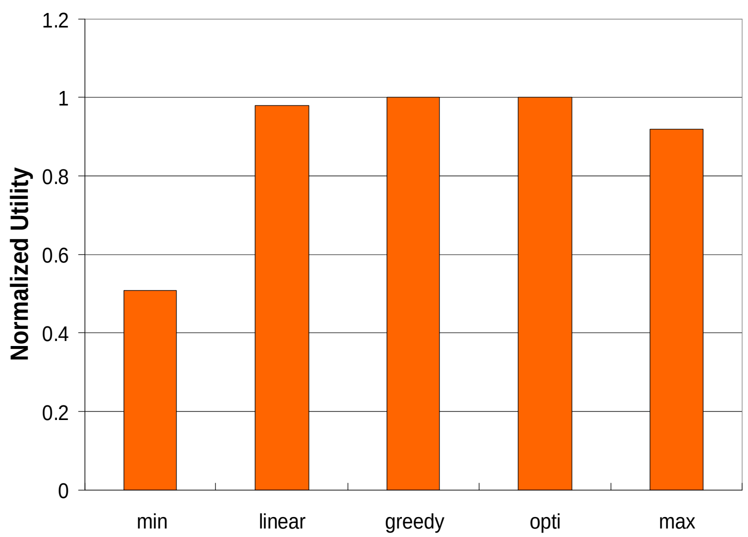
Simulation

- Permits exploring a large multi-dimensional task set space
- Simulate various hardware configurations, RT scheduling, DVS mechanisms
  - Static RM, Static EDF, ccRM, ccEDF, laEDF
- Generated 1000 random task sets, each with 10 tasks, and each of which has up to 5 QoS levels
  - QoS degradation models period extension, imprecise computation, algorithmic change

Simulation Results

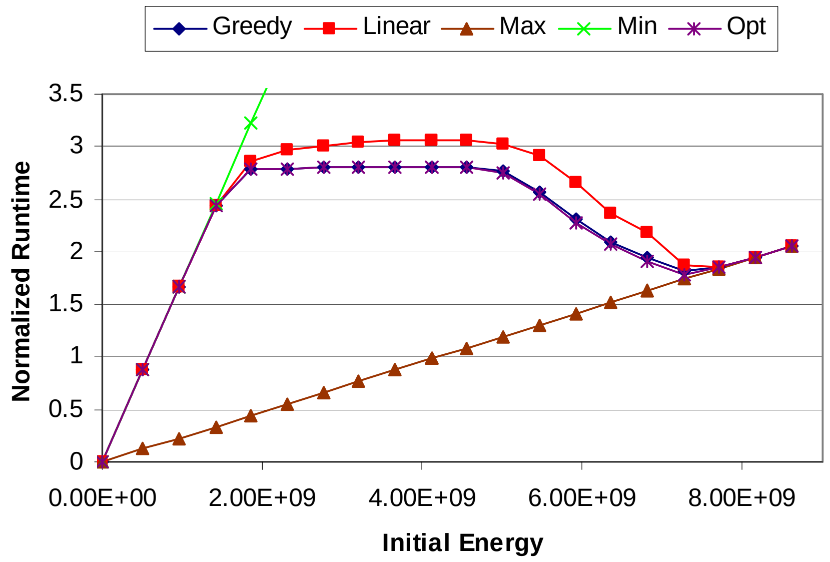
- EQoS algorithms w/o DVS achieve desired runtime
- DVS conserves extra energy, throws off estimated runtime





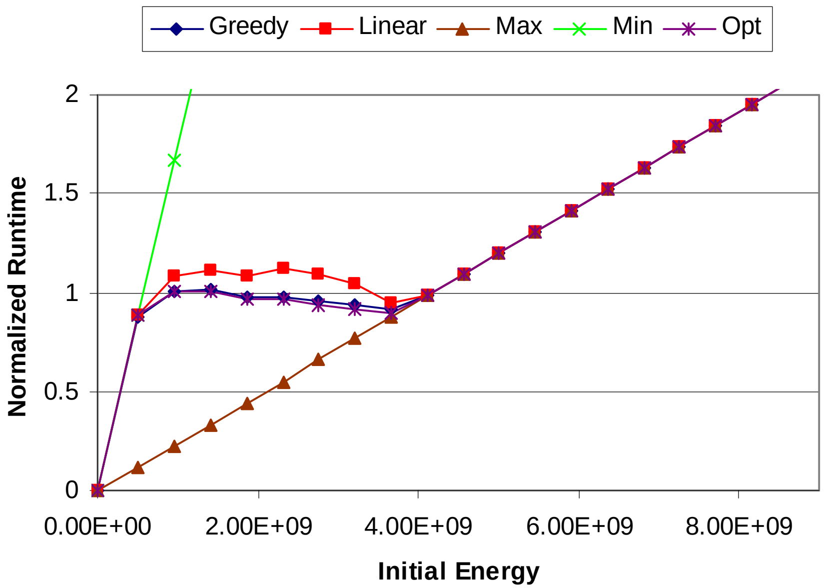
Simulation Results - DVS

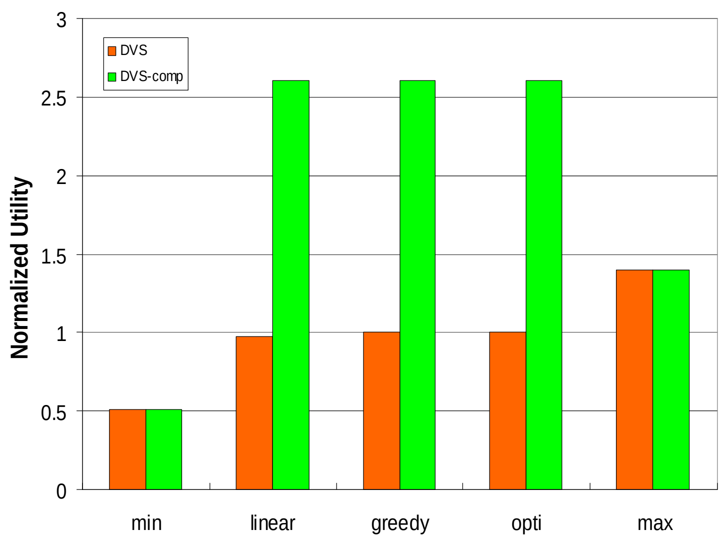
- DVS increases energy efficiency
- Throws off adaptation -- extends runtime
- 3 volt/freq:
  - 5V, 1.0*fmax
  - 4V, .75*fmax
  - 3V, .35*fmax



Simulation Results, cont•d

- DVS compensation achieves desired runtime with higher utility



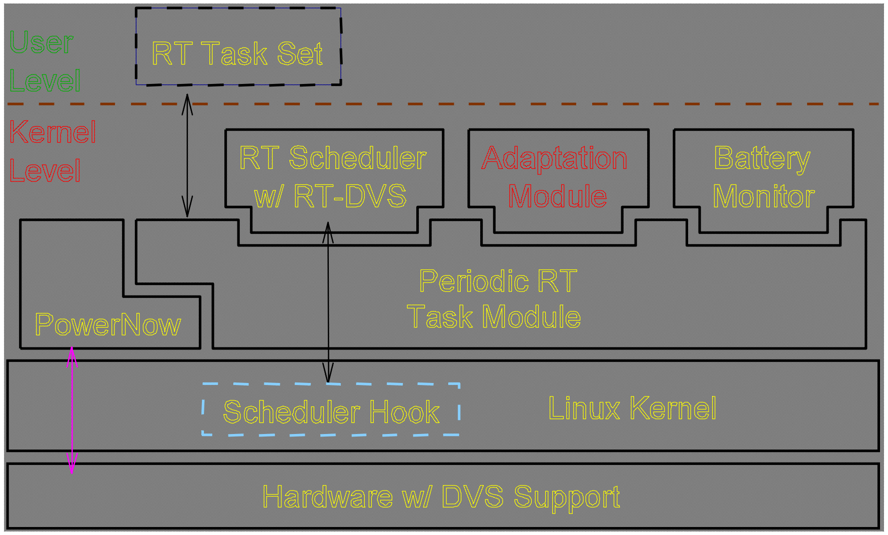


Adaptation w/ DVS Compensation

Utility comparison between DVS compensation and w/o compensation

Implementation

- Implemented on Linux 2.2
  - Periodic real-time support
  - PowerNow! driver
  - Real-time scheduler modules
  - EQoS adaptation module
  - Battery monitoring module



- Currently supports Athlon, Duron, K6-2 processors that implement AMD•s PowerNow! Technology

Experiments

- Measurements on a Compaq Presario 1200Z
- Implement RT version of Lame MP3 encoder
  - use quality parameter to vary QoS
  - multiple concurrent instances
- Results follow trend observed in simulations

Conclusions

- RT-DVS provides low-level CPU voltage control
  - Maintains timing guarantees for RT tasks
  - Significant energy savings, comparable to non-RT DVS
- EQoS provides task/app adaptation in energy-constrained real-time systems
  - Provides guidelines to best utilize available energy among tasks
  - Frames energy adaptation as a tractable problem
  - Heuristics work nearly as well as optimal algorithms in practice

Ongoing and Future Work

- Fine-grained measurement of energy consumption and its feedback to EQoS manger
- New task model based on energy consumption
- Energy consumption by components other than CPU, such as memory, flash memories/micro-disks, communication procotols (IEEE 802.11 and other sensor networking protocols)
- Construction of and experimentation with a network of iPaqs.

Algorithms -- Summary

- Optimal solutions: dynamic programming (DP), branch and bound (BB)
  - NP-hard
  - DP high memory overhead, runtime overhead
  - BB exponential upper bound on computation
- Heuristics: LMCKP, Greedy
  - Under utilizes power budget
  - Very fast computation
  - Greedy still close to optimal