

---

# Practical Considerations for Power-Aware Real-Time Scheduling

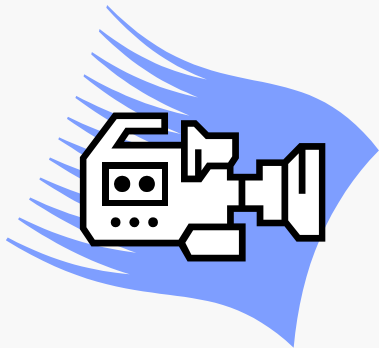
*X. Sharon Hu*

*Dept. of Computer Science and Engineering  
University of Notre Dame, Indiana*

**Joint work with Bren Mochocki (Notre Dame) and Gang Quan (U. South Carolina)**

# Why Power-Aware Real-Time Scheduling?

- Real-time embedded systems
  - Provide timely and predictable services
  - Power concerns due to
    - Limited energy sources
    - Expensive cooling costs
- Goal: meeting time requirements with minimal power/energy



# Presentation Outline

---

- Background
- Problem definition
- Voltage/speed scheduling under the Earliest Deadline First (EDF) policy
- Voltage/speed scheduling under the Fixed Priority (FP) policy
- Experimental results
- Conclusions and future work

# Real-Time (RT) Embedded Systems

---

- A single processor executing a set of tasks
- Tasks can be periodic or sporadic
- The CPU must be able to handle peak requirements
- Tasks are prioritized according to
  - Earliest Deadline First (EDF), or
  - Fixed Priority (FP)
- Preemption is allowed

# CMOS Technology

## ■ Power in CMOS Circuits:

$$P = \underbrace{k_2 C_L V_{dd}^2 f}_{\text{Switching power}} + \underbrace{I_{leak} V_{dd}}_{\text{Leakage power}} + k_1 I_{direct} V_{dd}$$

## ■ Delay in CMOS Circuits:

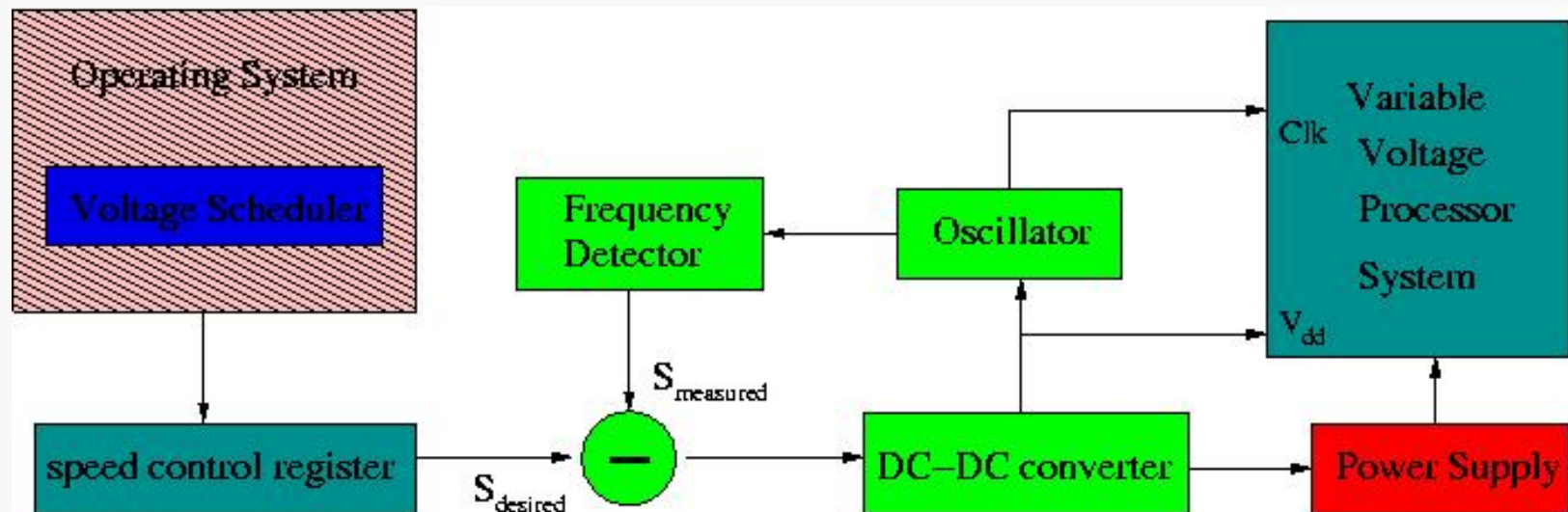
$$t_p = k \frac{V_{dd}}{(V_{dd} - V_{th})^\alpha}, \alpha = [2,3]$$

## ■ Power is a convex function of clock frequency (speed)

# Reducing Switching Power/Energy

## ■ Dynamic supply Voltage Scaling

- Change a processor's supply voltage and speed during runtime
- Increase energy efficiency by matching the processor's performance to the current workload
- Need a "good" voltage schedule



# Reducing Leakage Power/Energy

---

## ■ Dynamic $V_{th}$ scaling

- Change a processor's threshold voltage during runtime
- Reduce active mode leakage
- Can be combined with dynamic Vdd scaling
- **Need a “good” voltage schedule**

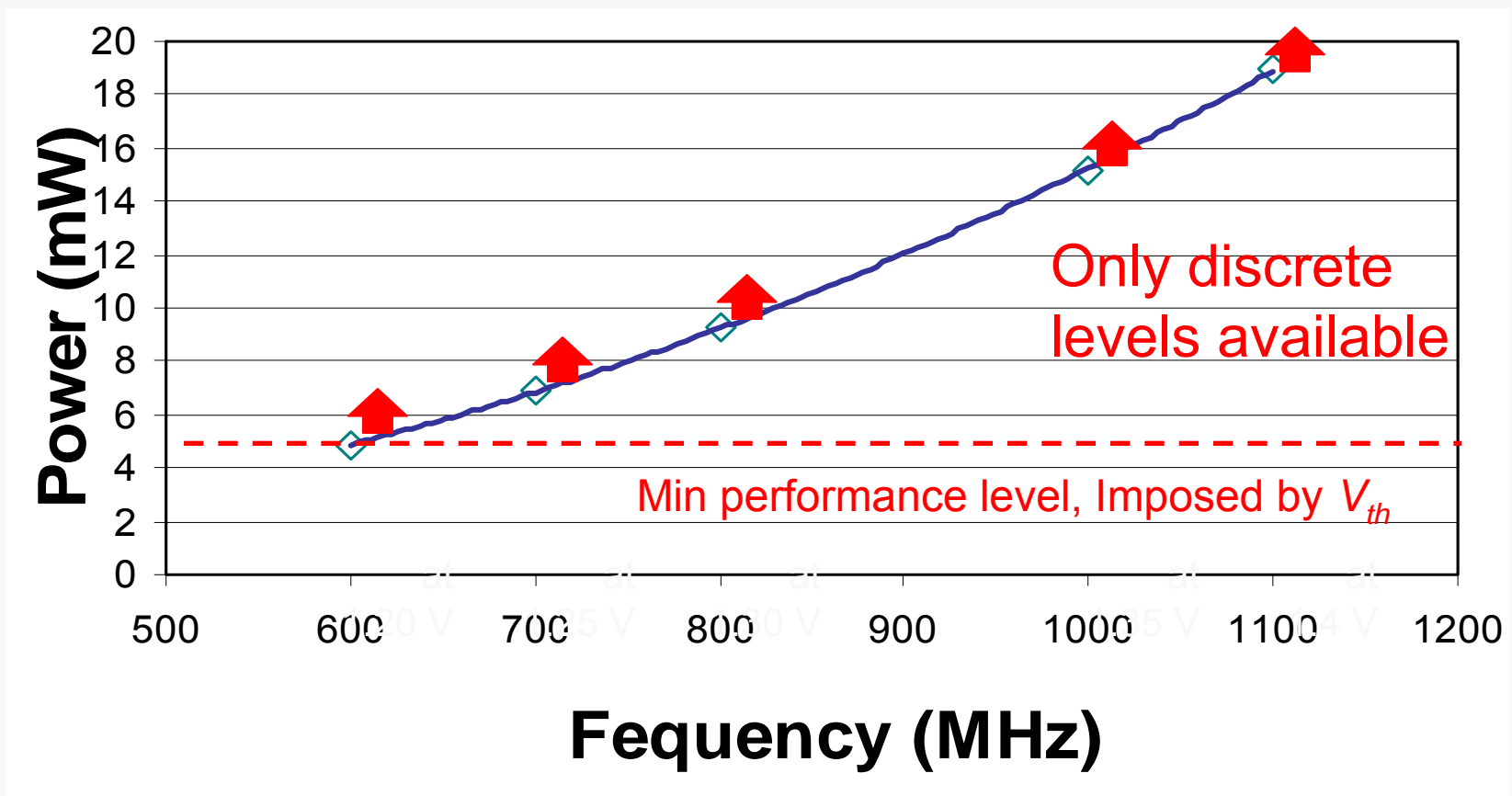
## ■ Low power standby

- Put processor in low-leakage mode when idle
- **Need a “good” schedule** since the mode change is not instantaneous

# Non-Ideal Features of DVS Processors

## ■ Power v.s. frequency curve of AMD Mobile Athlon4

$$P(f) = 2 \times 10^{-5} f^2 - 7.3 \times 10^{-3} f + 1.7664$$



# Transition Overhead

Technology	Time Overhead ( $\Delta t$ )	E Overhead ( $\Delta E$ )
LART ARM SA-1100 (Delft University)	PLL – 140 $\mu$ S DC-DC -Converter increase 40 $\mu$ S (max) decrease 5.5 mS (max)	14 $\mu$ J
Mobile Athlon 4	100 $\mu$ S	240 $\mu$ J
	Commercially Available	
ARM8 Core (UC-Berkeley)	26 $\mu$ S (max)	6.4 $\mu$ J (max)

- Can be even more significant when synchronization with external components is needed

# Problem Definition

- A set of independent, RT Jobs:  $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ 
  - release times  $r_i$
  - deadlines  $d_i$
  - worst case execution cycles  $c_i$
- A processor
  - Constant Time Transition Overhead:  $\Delta t$
  - Energy Transition overhead:  $\Delta E(V_1, V_2)$
  - A set of valid speeds (normalized):  $S = \{S_1, S_2, \dots, 1.0\}$
  - A set of corresponding voltages:  $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$
  - Instructions are not executed during speed transitions
  - A breakeven interval (for low power mode):  $\Phi$
- **Goal:** Produce a valid schedule
- **Secondary Goal:** Minimize Energy

# Previous Work (1)

---

- A large body of work on DVS scheduling but they
  - either ignore all limitations (theoretical papers), e.g.,
    - **EDF policy**: Yao, Demers and Shenker (FOCS'95), Aydin, Melhem, Mosse and Alvarez (RTSS'01)
    - **FP policy**: Shin and Choi (DAC'99), Pillai and Shin (SOSP'01), Quan and Hu (DAC'01)
    - Resulting schedules can be invalid
  - or account for one of the three, without considering the others, e.g.
    - Kwon and Kim (DAC'03) consider discrete voltage levels
    - Saewong and Rajkumar (RTAS'02) consider large transition overhead
    - Doing so can have hidden side-effects

# Previous Work (2)

---

- Limited amount of work on combined leakage and switching energy reduction through scheduling
  - EDF policy:
    - Irani, Shukla and Gupta (SODA'03): Fundamental observations
    - Jejurikar, Pereira and Gupta (DAC'04): Assign a single procrastination interval to each task
  - FP policy:
    - Jejurikar and Gupta (LCTES'04): Assign a single procrastination interval to each task
  - Key is to find maximal tolerable delay for each task instance
  - Each task instance and aperiodic job can have its own slowdown factor and thus procrastination interval

# Our Results

---

- **For EDF-based and FP-based scheduling**
  - Basic, offline algorithms that guarantee to produce valid schedules under time transition overhead
  - Unified, offline algorithms that account for all three limitations
  - Algorithms to compute the “latest start time” (i.e., the procrastination interval) of each task instance or job
- **For FP-based scheduling**
  - An online algorithm to account for all three limitations

# Presentation Outline

---

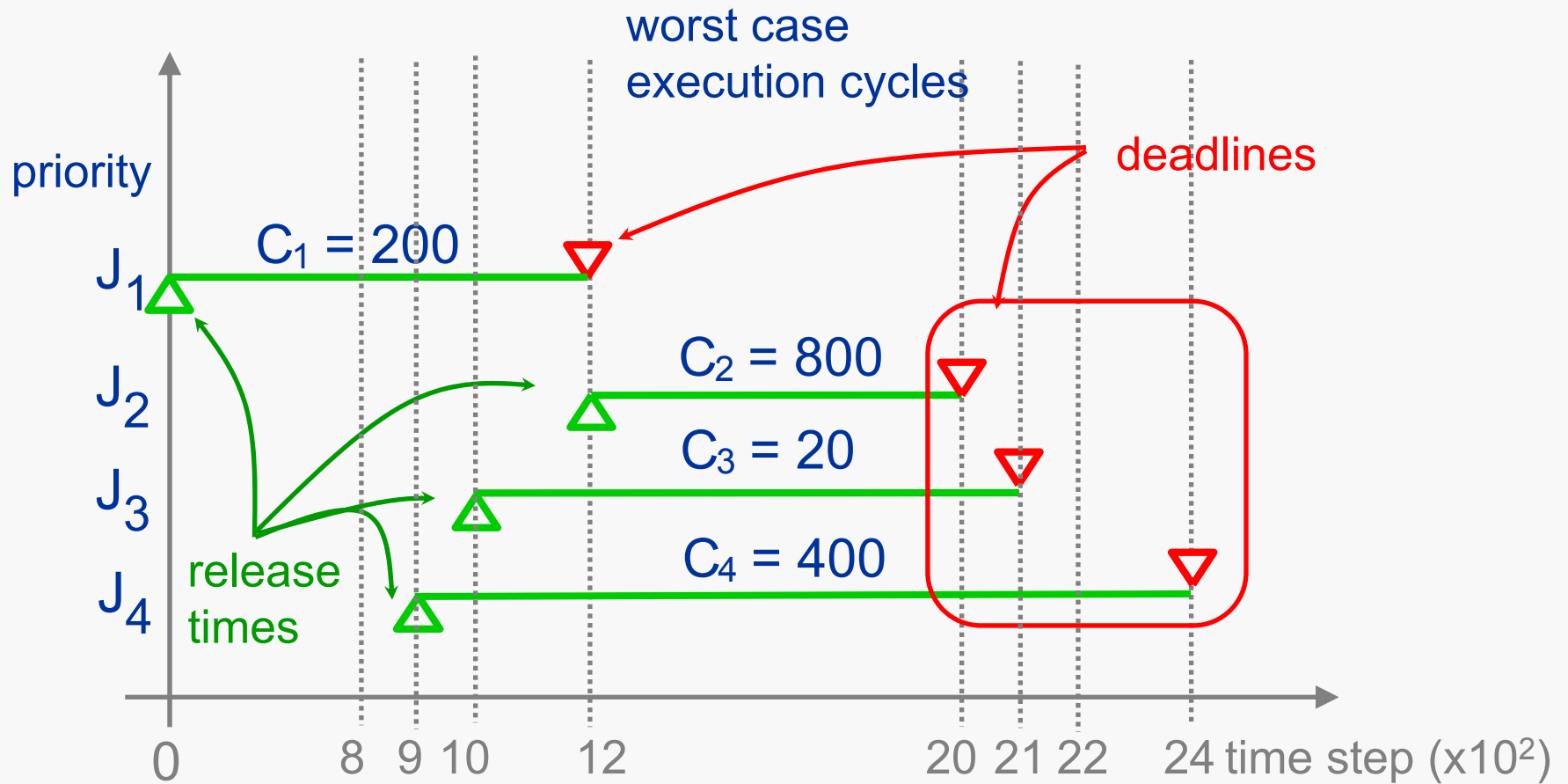
- Background
- Problem definition
- Voltage/speed scheduling under the Earliest Deadline First (EDF) policy
- Voltage/speed scheduling under the Fixed Priority (FP) policy
- Experimental results
- Conclusions and future work

# Low-Power EDF (LPEDF) (1)

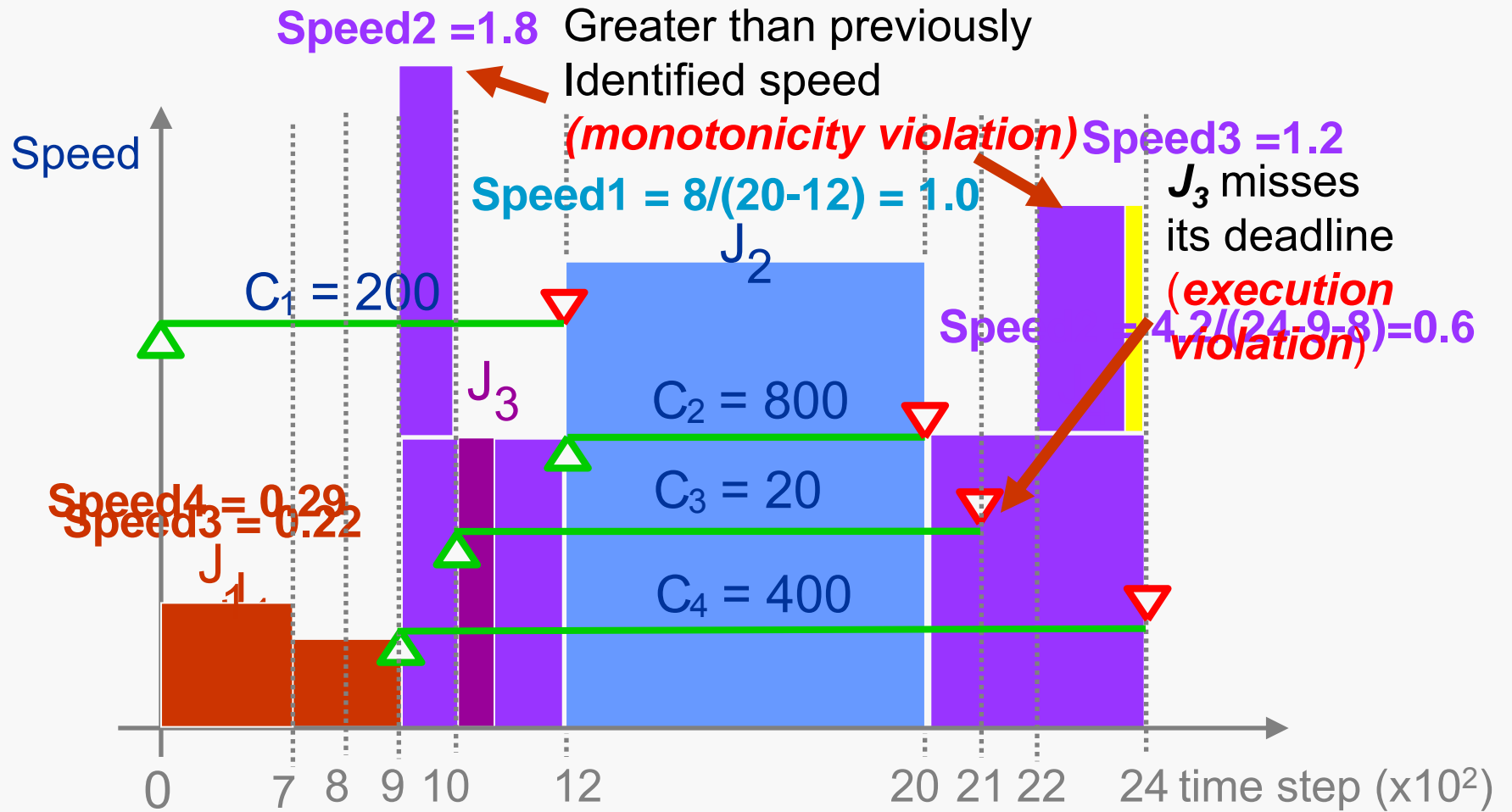
---

- Yao, Demers, and Shenker, FOCS 1995
- Optimal for an ideal processor (i.e., a continuous voltage/speed processor without transition overhead)
- Iteratively finds the most demanding interval, called critical interval, and assigns a corresponding voltage/speed level
- Time Complexity:  $O(n^3)$ ,  $n = \# \text{ of jobs}$
- Advantages
  - Optimal Off-line Voltage Schedule
  - Can be applied online
- Disadvantages
  - Assumes an ideal DVS processor

# Low-Power EDF (LPEDF) (2)



# Low-Power EDF (LPEDF) (3)



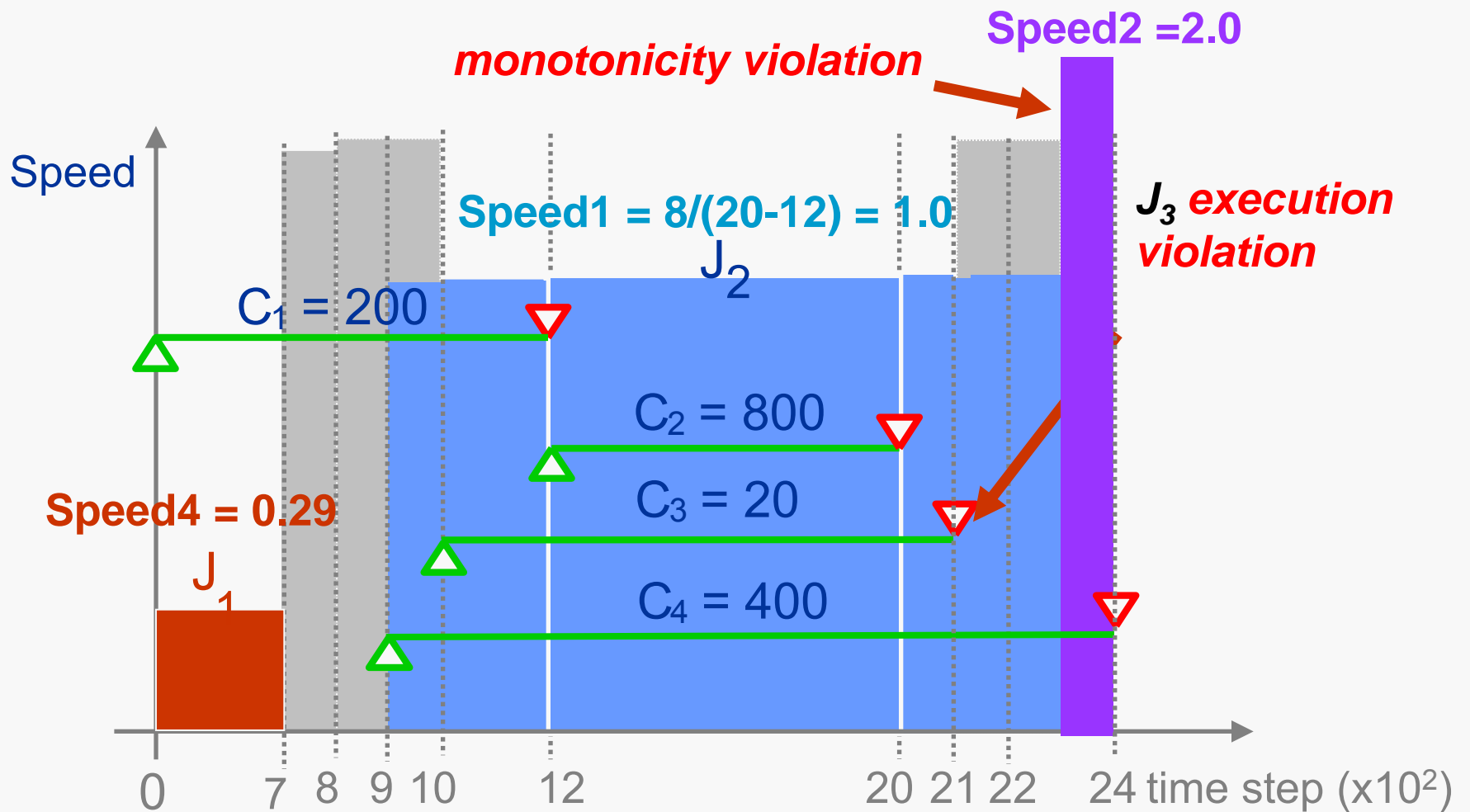
Impact of Time Overhead ( $\Delta t=200$ )

# Time Overhead EDF (TOEDF) (1)

---

- **To Meet all deadlines**
  - Remove  $[t_a - \Delta t, t_b + \Delta t]$  instead of just  $T_c = [t_a, t_b]$
- **To handle *Monotonicity Violations***
  - Extend previous critical interval  $T_{i-1}$  to include all jobs in the overshoot interval  $T_i$
  - $T_{i-1}$  and  $T_i$  will be adjacent intervals
- **To handle *Execution Violations***
  - Extend  $T_c = [t_a, t_b]$  iteratively to include all jobs in  $[t_a - \Delta t, t_b + \Delta t]$  until no jobs exist in  $[t_a - \Delta t, t_b + \Delta t]$

# Time Overhead EDF (TOEDF) (2)

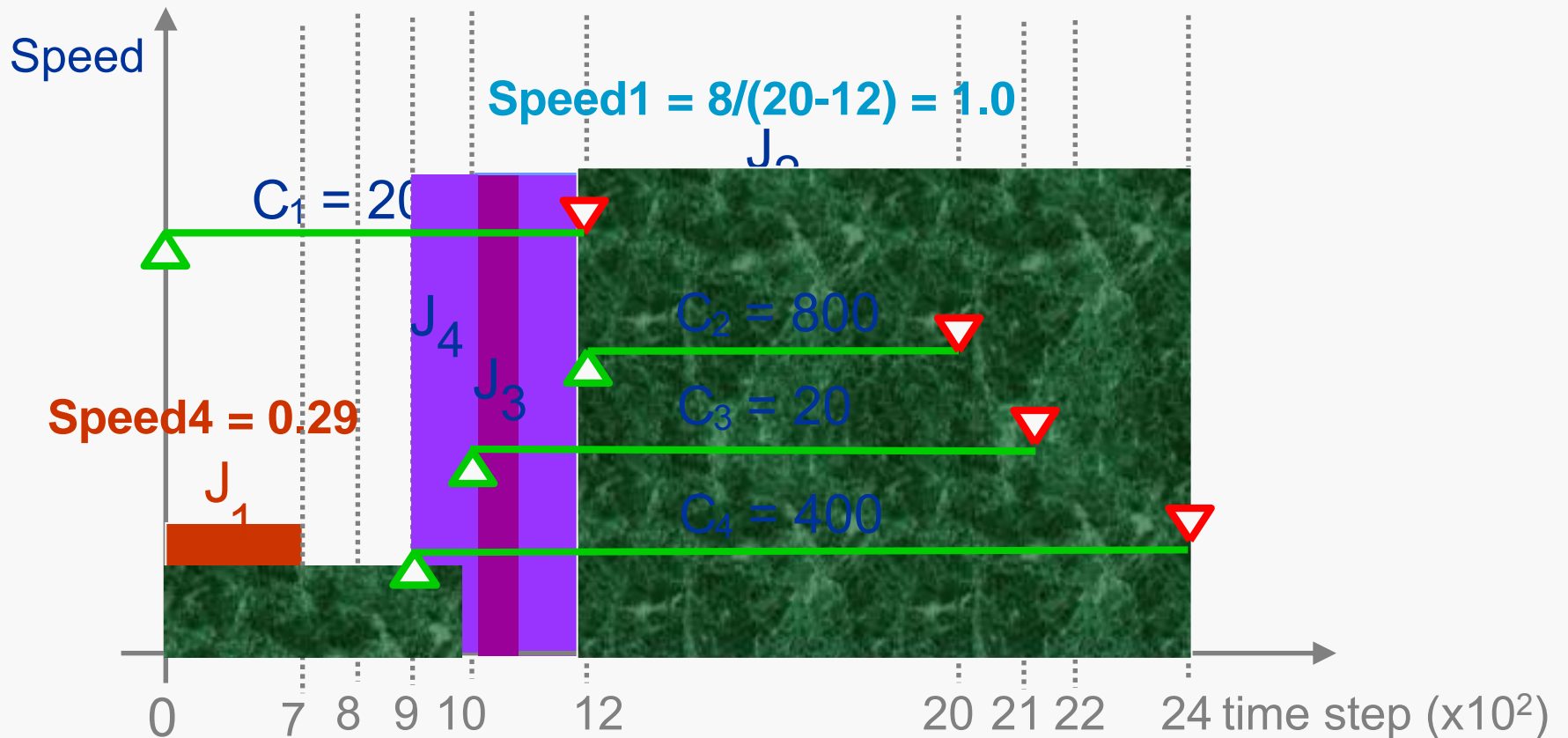


# Time Overhead EDF (TOEDF)

---

- **Time Complexity:**  $O(n^3)$ ,  $n = \# \text{ of jobs}$
- **Advantages**
  - Schedule is always valid
  - Simple to implement
  - Only a small constant factor more complex than LPEDF
- **Disadvantages**
  - Rounds up to match discrete levels
  - Doesn't consider energy overhead
  - Slack can exist in critical intervals

# TOEDF can be inefficient



- We want to find a *minimum length* interval that will finish all jobs by their deadlines

# Latest Start Time

- $t_{LS}(i)$  is the latest start time for job  $J_i$

$$t_{LS}(i) = \underbrace{d_i}_{\text{Time available to } J_i} - \underbrace{\sum_{j=1}^i \frac{c_j}{s^*}}_{\text{Time taken by higher priority jobs}}$$

- $s^*$  is a speed at which all jobs in  $\mathcal{J}$  can be executed and still meet their respective deadlines
- Each job can also have its own speed

- *The job set  $\mathcal{J}$  scheduled by EDF has the latest start time,  $t_{LS}$ :*

$$t_{LS} = \min\{t_{LS}(i) \mid i = 1, 2, \dots, |\mathcal{J}|\}$$

# $t_{LS}$ yields a Minimum Interval

---

- Using  $t_{LS}$  as the starting time always yields a minimum interval for  $\mathcal{J}$
- The minimum interval is used to include *monotonicity* and *execution violation* jobs in the previous critical interval
- Other minimum intervals exist that reduce the energy more in some cases (future work)

# Discrete Voltage Levels

---

- TOEDF will just round up (inefficient)
- AllocVT (Kwon and Kim, DAC'03) also becomes inefficient when  $\Delta t$  is not negligible
- Our approach
  1. Round speed up
  2. Select a busy interval to keep
  3. Reschedule remaining jobs

# Energy Overhead ( $\Delta E$ )

---

- TOEDF does not consider  $\Delta E$
- Neighboring intervals can be combined, if the combination reduces energy consumption
- The minimum interval is used to execute jobs in neighboring intervals at one speed

# Unified Algorithm EDF (UAEDF)

---

- **Combines the techniques to handle time/energy transition overhead and to match discrete voltage levels**
  - Find minimum interval to handle violations
  - Avoid costly transitions (due to energy transition overhead) using the minimum interval.
    - **During algorithm execution or post processing**
  - Explicitly handle discrete levels
- **Keeps the  $O(n^3)$  time complexity**

# Presentation Outline

---

- Background
- Problem definition
- Voltage/speed scheduling under the Earliest Deadline First (EDF) policy
- **Voltage/speed scheduling under the Fixed Priority (FP) policy**
- **Experimental results**
- **Conclusions and future work**

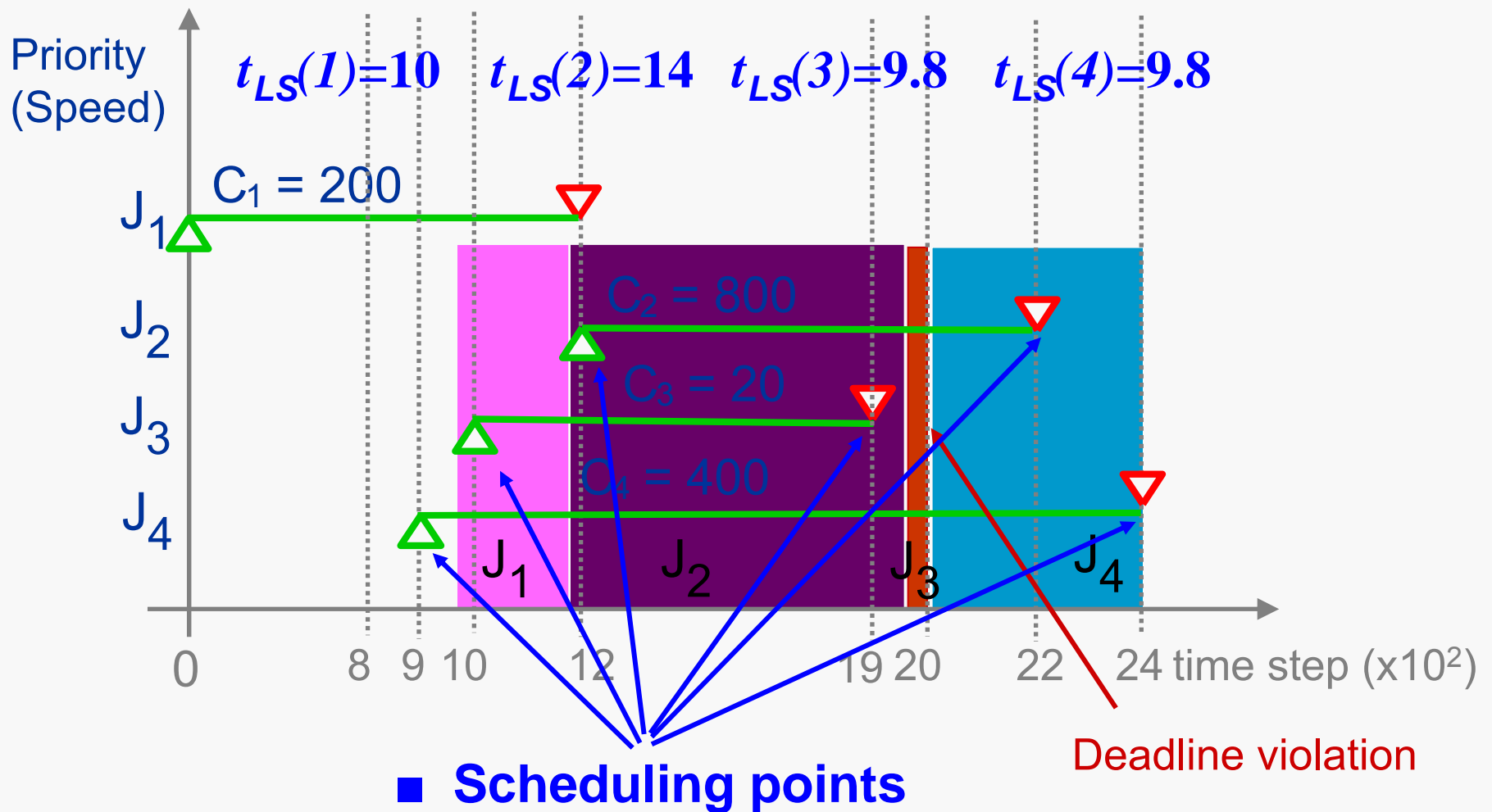
# Fixed-Priority Scheduling

---

- Finding the optimal voltage schedule is NP hard (Yun and Kim, ACMTECS'03)
- We proposed the essential-interval based method (Quan and Hu, DAC'01)
  - Find the **minimum speed** needed to execute each job
  - Take the interval corresponding to the **maximum** of the minimum speeds as the **critical interval**
  - Is an efficient heuristic ( $O(n^3)$ )
  - Can be extended to handle the transition overhead similar to LPEDF

# Why Is It Hard for FP Scheduling

$S^* = 1.0$



# Latest Start Time for FP Scheduling (1)

- The latest start time for job  $J_i$

$$t_{LS}(i) = \max \left\{ t - \sum_{j=1, r_j < t}^i \frac{C_j}{S^*} \mid t \in SP(J_i) \right\}$$

- Considers all the eligible scheduling points
- Guarantees  $J_i$  not missing its deadline
- $s^*$  is a speed at which all jobs in  $\mathcal{J}$  can be executed and still meet their respective deadlines
- Each job can also have its own speed

# Latest Start Time for FP Scheduling (2)

---

- The latest start time of job set  $\mathcal{J}$ :

$$t_{LS} = \min\{t_{LS}(i) \mid i = 1, 2, \dots, |\mathcal{J}|\}$$

- *The latest start time gives the minimum interval*
- *Can be determined either offline or online*

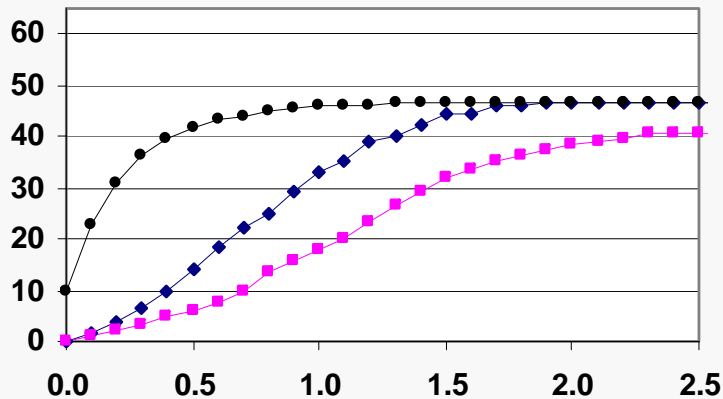
# Presentation Outline

---

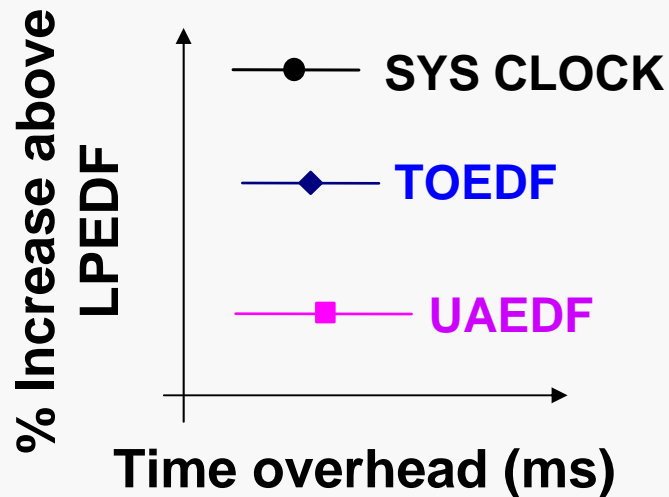
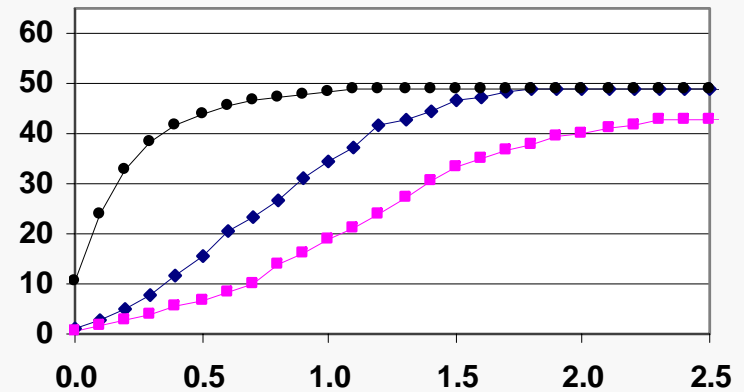
- Background
- Problem definition
- Voltage/speed scheduling under the Earliest Deadline First (EDF) policy
- Voltage/speed scheduling under the Fixed Priority (FP) policy
- **Experimental results**
- **Conclusions and future work**

# Randomly Generated Job Sets

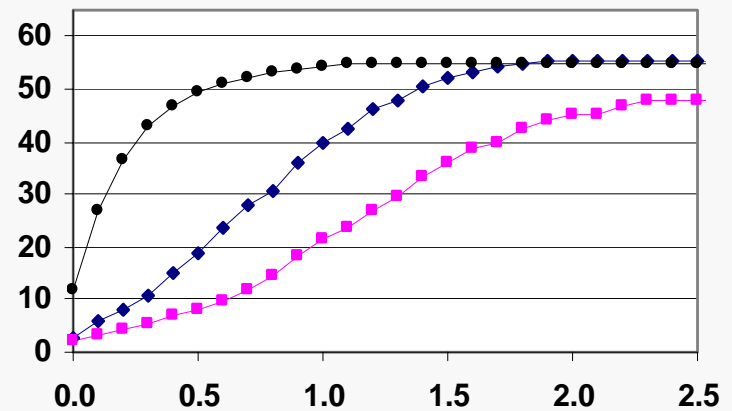
(a) Continuous Voltage Levels



(b) 14 Voltage Levels

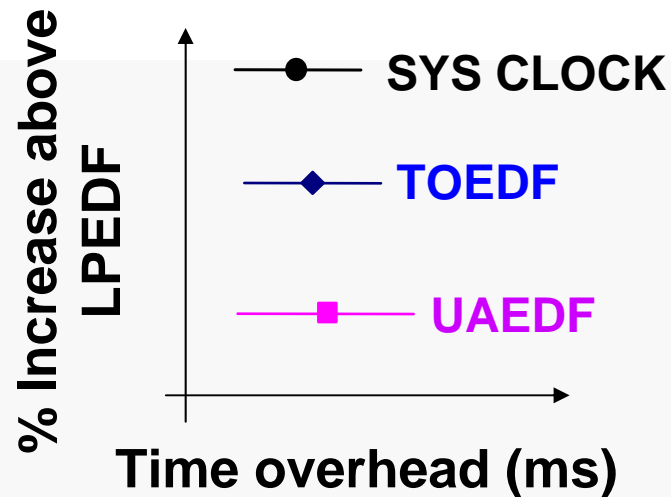
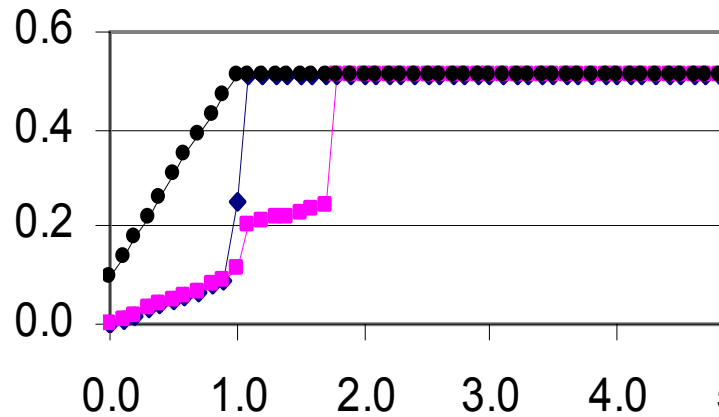


(c) 5 Voltage Levels

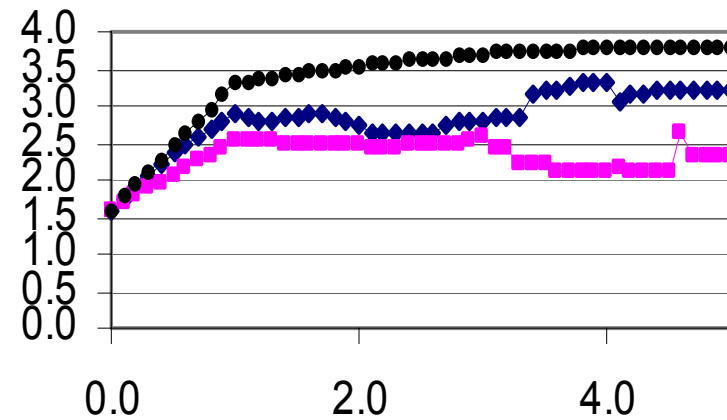


# Video Phone Job Set

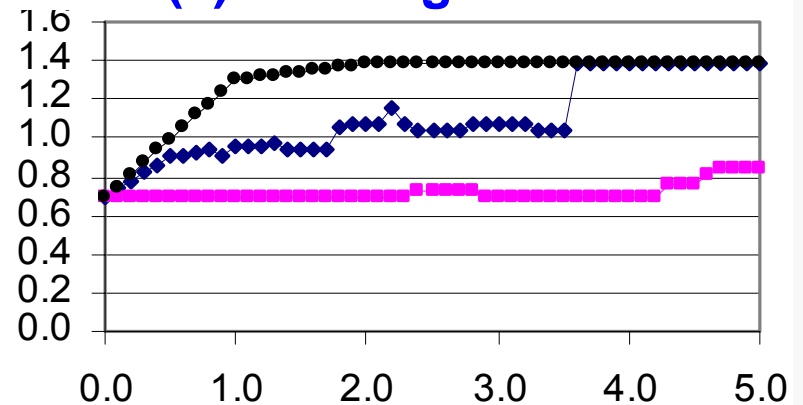
(a) Continuous Voltage Levels



(b) 14 Voltage Levels

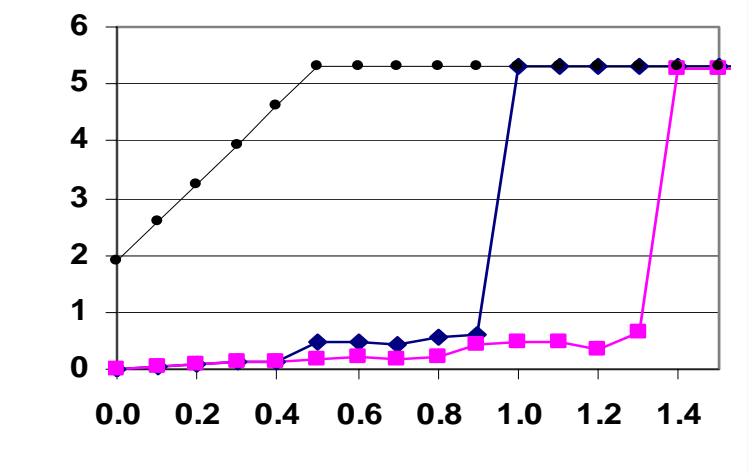


(c) 5 Voltage Levels

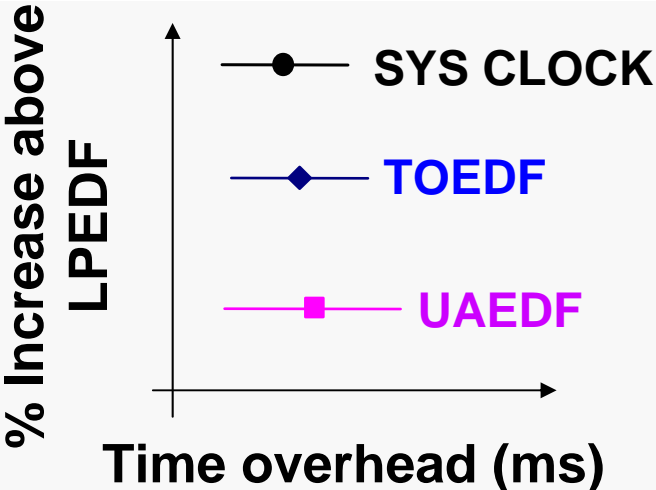
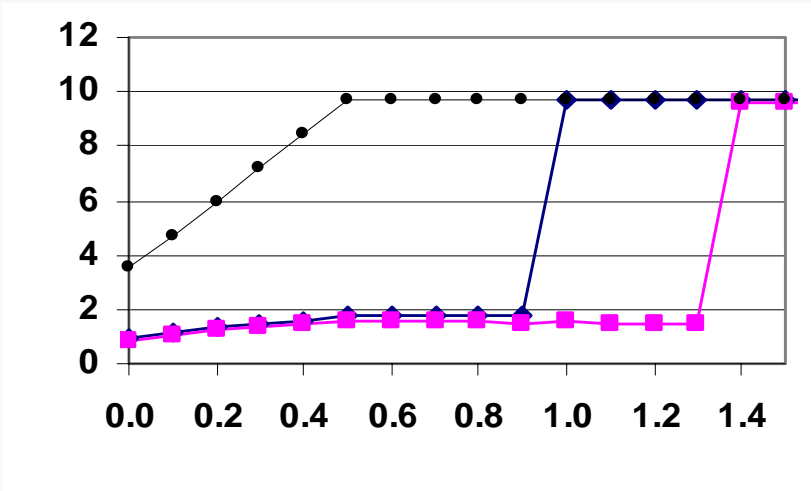


# Avionics Job Set

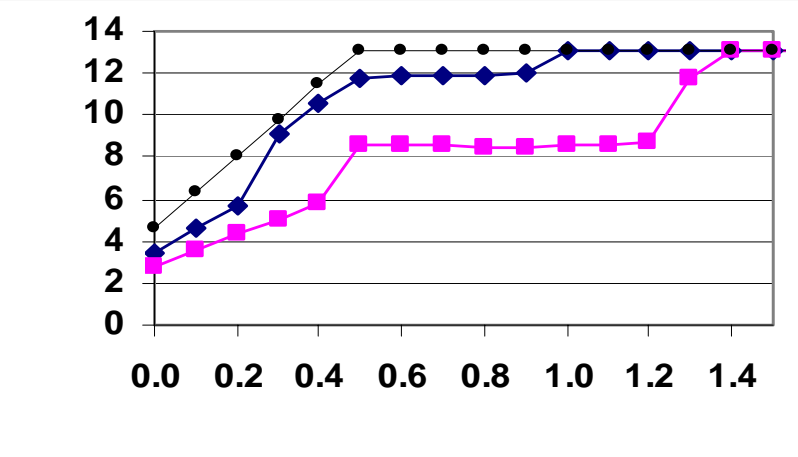
(a) Continuous Voltage Levels



(b) 14 Voltage Levels



(c) 5 Voltage Levels



# Conclusions

---

- Time and energy overhead can impact both the feasibility and energy consumption if not handled carefully
- We developed a set of algorithms to deal with practical limitations on DVS processors
- Our algorithms for finding the latest start times can provide more opportunities for saving leakage energy

# Future Work

---

- **Modify algorithms to work with different processor models (e.g. variable  $\Delta t$ , instructions execute during transitions, etc...)**
- **Investigate the optimality property of DVS processors with non-negligible transition overheads**
- **Extend the work to multiple processor and SoC platforms**