CS343: Operating System

Scheduling Algorithms

Lect13 : 25th Aug 2023

Dr. A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Quiz 1

- [5] What is system call and how it is different from API call? What kind of services the OS provide by sys_calls?
- [4] Provide all the scenario, in which a process can enters to ready queue.
- [6] Given 5 jobs with arrival time 0, 1, 1, 3, 5 and execution time 2, 8, 9, 7, 5 respectively. Calculate average waiting time if scheduled using

  (a) FCFS (b) SRT, (c) RR with q=2
- [8] Given N independent jobs without pre-emption (a_i=0) with each job have some weight/price/priority w_i associated with it need to be executed on one processor and goal is to minimize $\sum w_i C_{i,}$ where C_i is completion time. **Find an optimal approach to solve the same.**

Scheduling Algorithms

- Recap
  - Energy Efficient with DPM and DVFS
  - Real time Scheduler
- Scheduling Algorithms: Theory Overview

Scheduling: When No Scarcity of CPU Resource

Multicore System

- CPU can run at different frequencies
  - DVFS : P=Ps+alpha f3
  - **Base frequency**, Turbo-frequencies
- CPU can have different state
- Intel i7-1265UL: 2 Perf Core and 8 Eff. Core
  - 15W, Base f= 1.8Ghz, 2.7Ghz/4.8Ghz for E/P cores
- Our Institute Insurance Policy (10000 Person)
  - Rs 2L Per person, 1Cr for maximum of two persons
  - If prob of sick with critical disease is 0.0002 → can I say it is good policy

Power Aware (PA) Computing

- Objective of PA computing/communications is
  - To improve power management and consumption
  - Using the awareness of power consumption of devices.
- Power consumption is most important considerations
  - In mobile devices due to limitation battery life.

Power Aware Computing

- System level power management
- Recent devices support multiple power modes.
  - CPU, disk, communication links, etc.
- **Resource Management and Scheduling Systems**
  - Can use these multiple power modes
  - To reduce the power consumption.

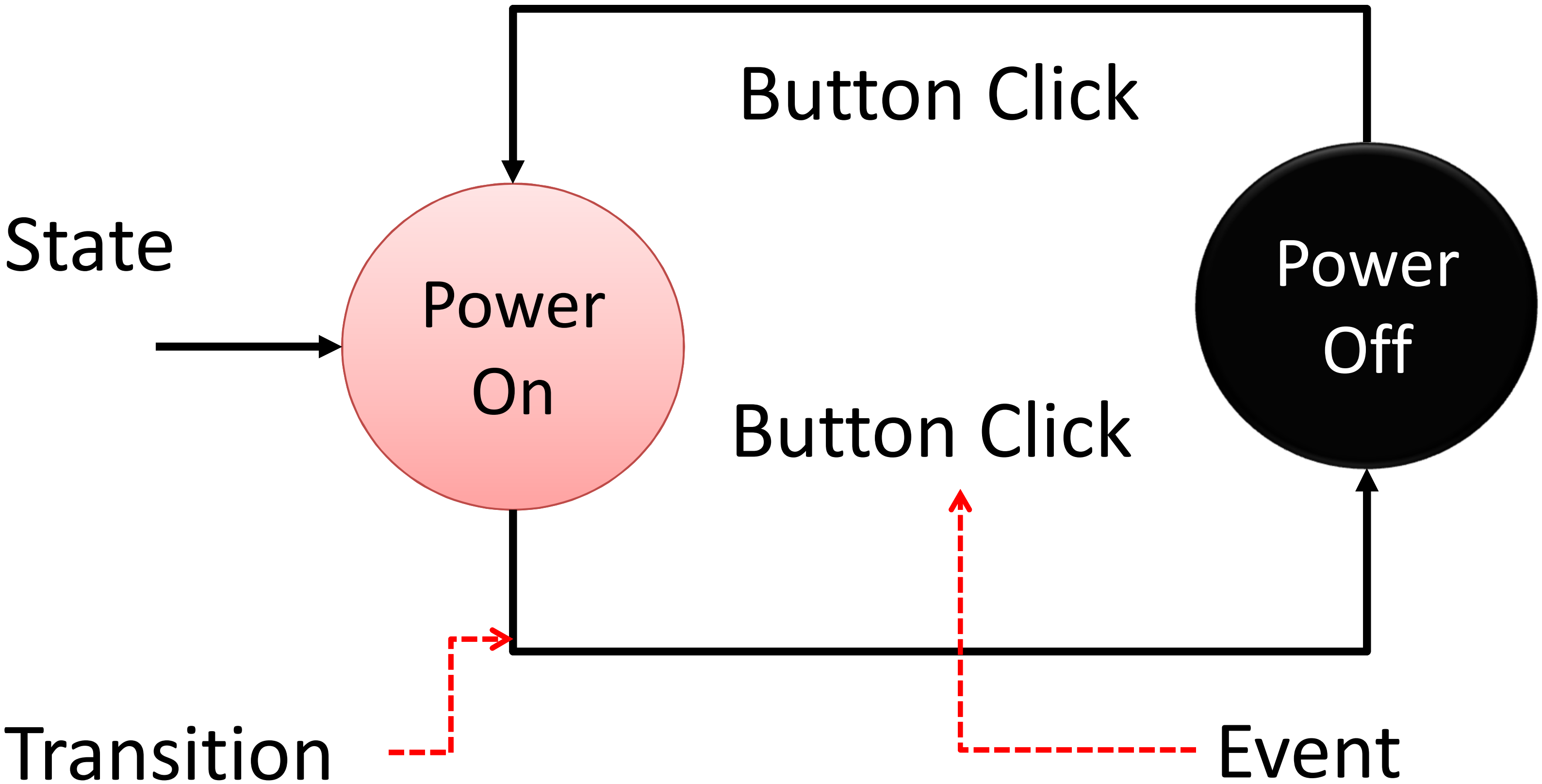
DVFS-based Power Aware Scheduling : Motivation

- Develop Resource Management and Scheduling Algorithms
  - That aim at minimizing the energy consumption
  - At the same meet the job deadline.
- Exploit industrial move towards
  - Utility Model/SLA-based Resource Allocation for Cloud Computing

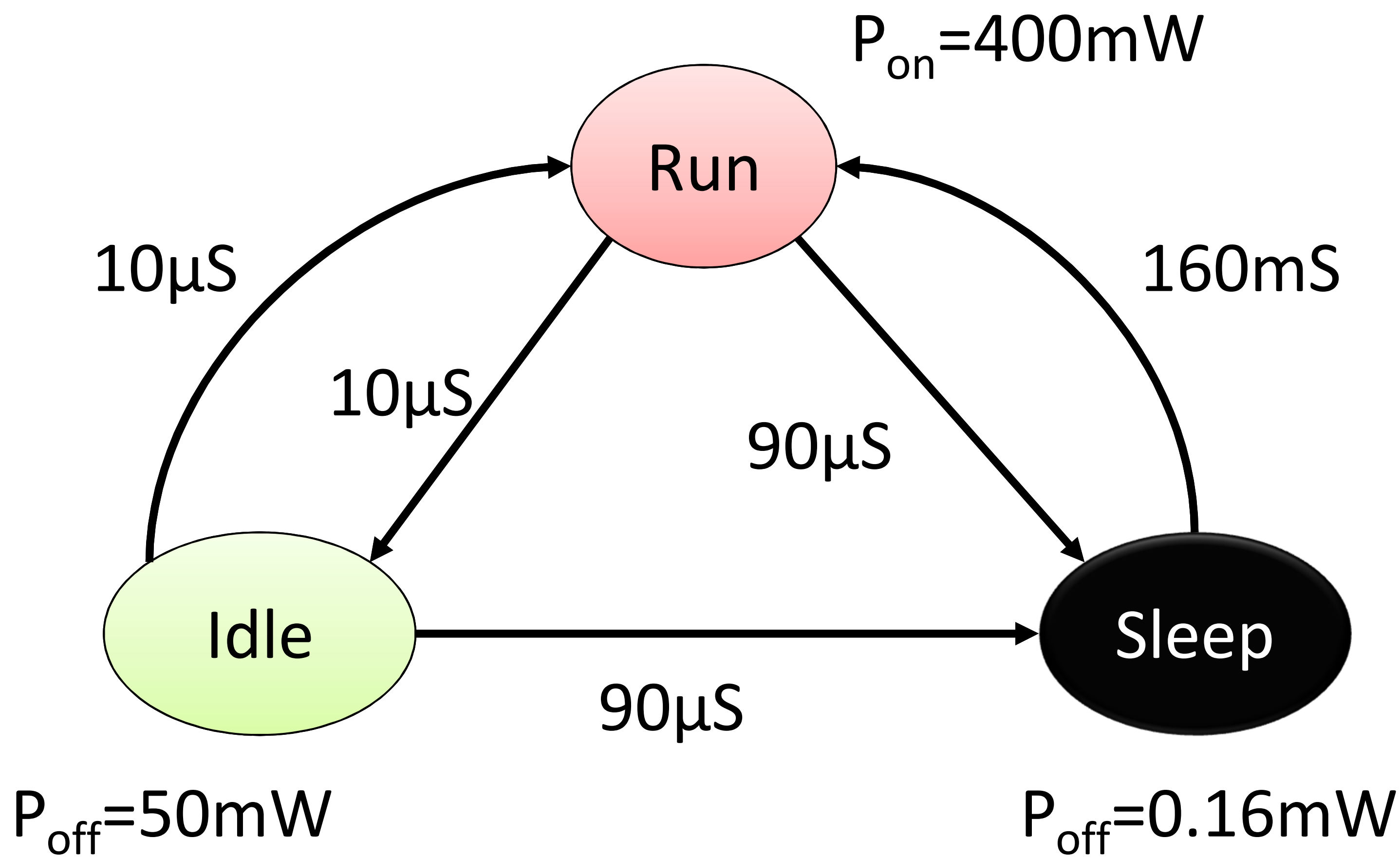
Static PM vs Dynamic PM

- Static PM
  - Invokes by user does not depends on user activities
  - Power down mode: c0, c1, …cm
    - Off, dose, nap, sleep, run
  - Mode exit upon receiving an interrupt
  - Power State machine
- Dynamic PM
  - Control power based on dynamic activity in CPU
  - Dynamically change freq, shut some parts
  - **Do when in Run State**

Static PM with Power States

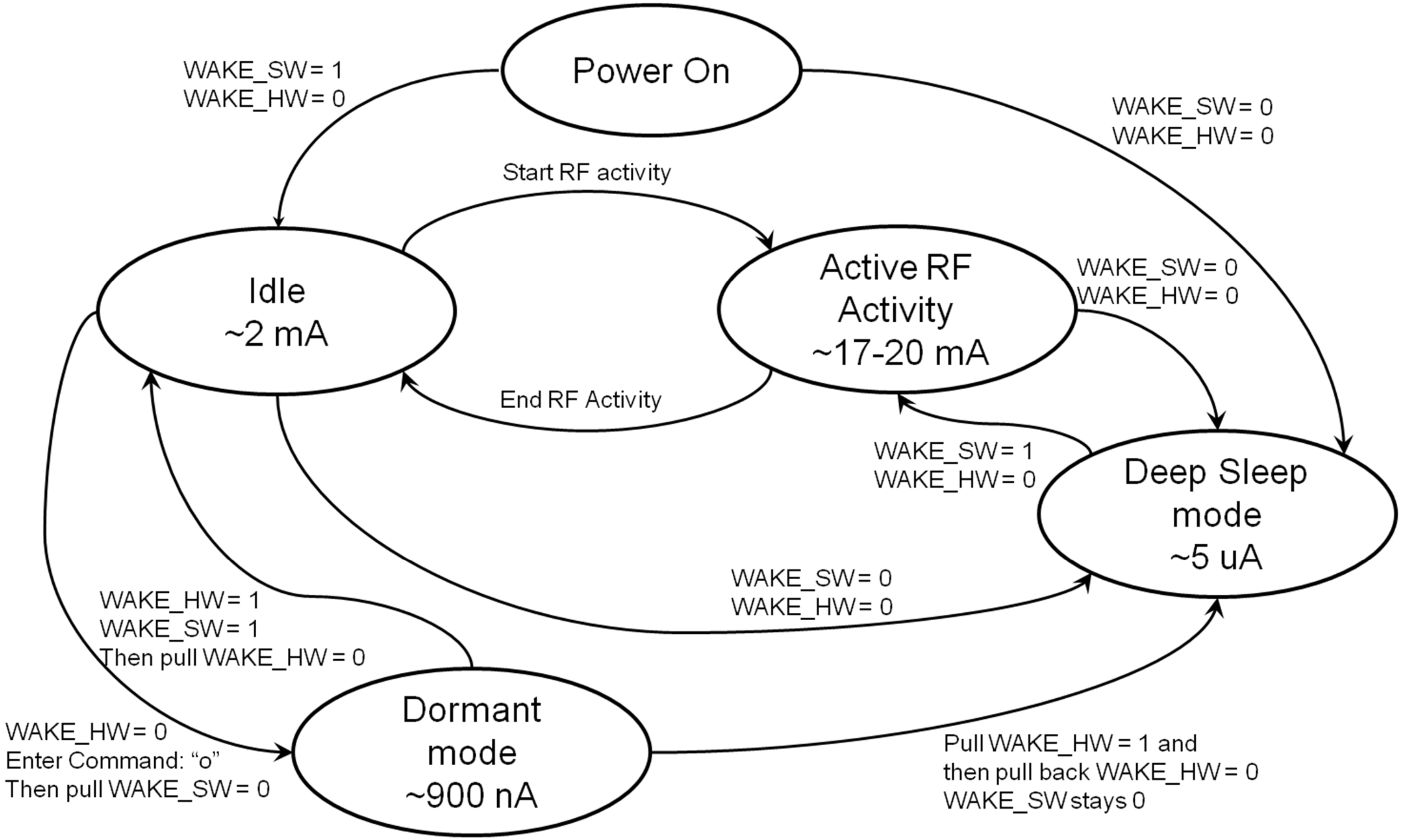


Static PM with Power States



During Transition $P_{TR}=P_{ON}$

Static PM with Power States

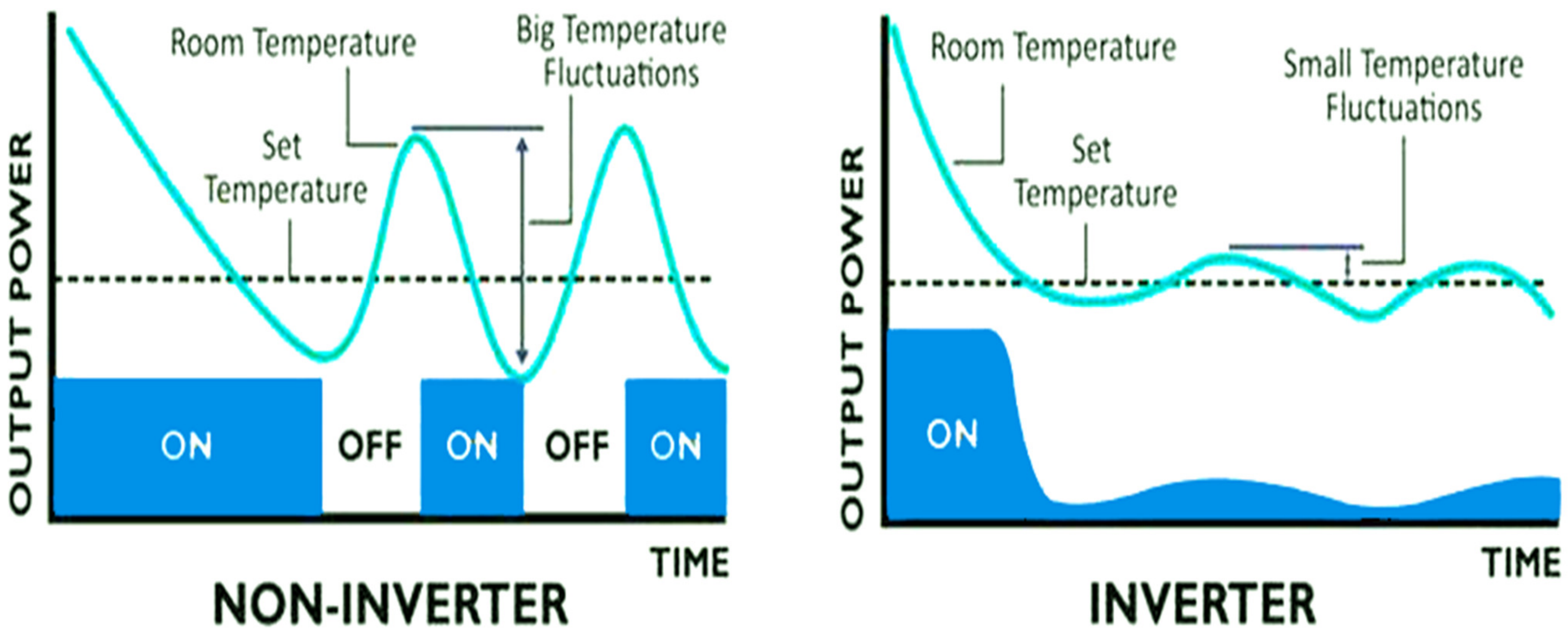


Real Life Issue: Inverter AC

- Inverter AC vs Non-Inverter AC
- Non-Inverter AC : Run fast and rest
- Non-Inverter AC: switch-of and switch-on mode
  - Sound, Fan on-off
- Inverter AC : Quit and required
  - Run at required speed : **Fun to compare with EMI**
  - Quieter than a mosquito

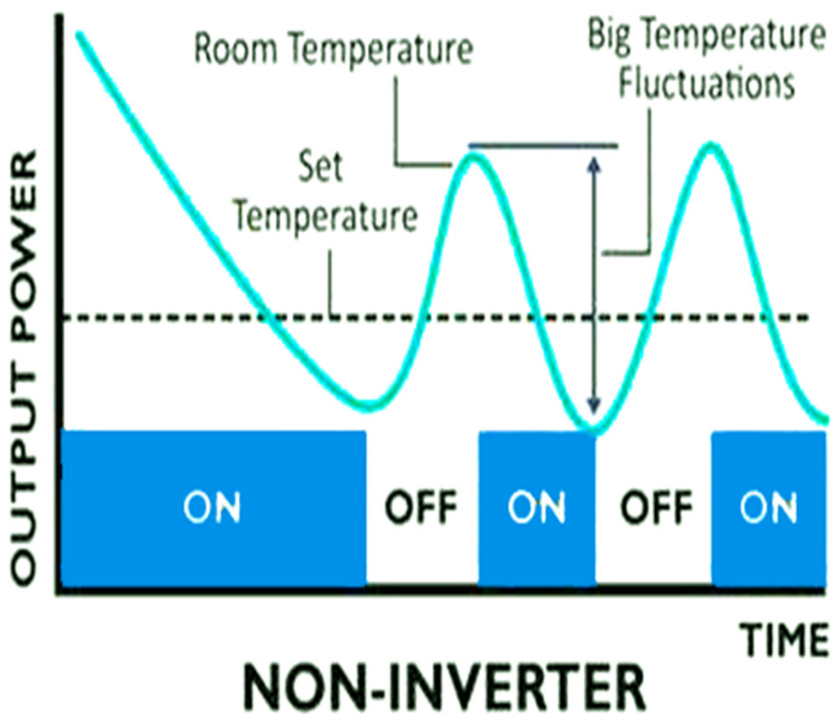
Real Life Issue: Inverter AC

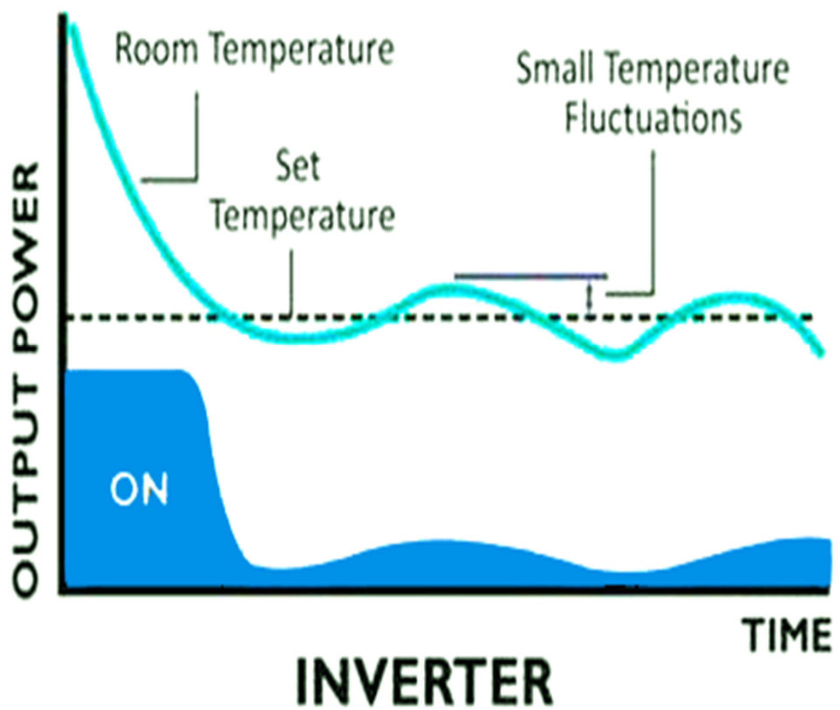
- Eco Friendly, less power consumption
- Makes little sound, Efficient Cooling/Heating
- No Voltage Fluctuation caused by compressor
- Can be run on solar panels



Real Life Issue: Inverter AC

- Suppose P is proportional to f^3
- Running at 1 speed and 0.5 speed
- DPM Running at 1 speed 50% time $E_{DPM}=1/2*(1)^3=1$
- And DVFS 0.5 speed all the time $E_{DVFS}=(0.5)^3=0.125$





DPM vs DVFS

- Inverter AC vs Non-Inverter AC
- Non-Inverter AC : Run fast and rest
- DPM : switch-of and switch-on mode
  - Sound, Fan on-off
- DVFS : Quit and required mode
  - Quieter than a mosquito
  - Run at required speed

DVFS

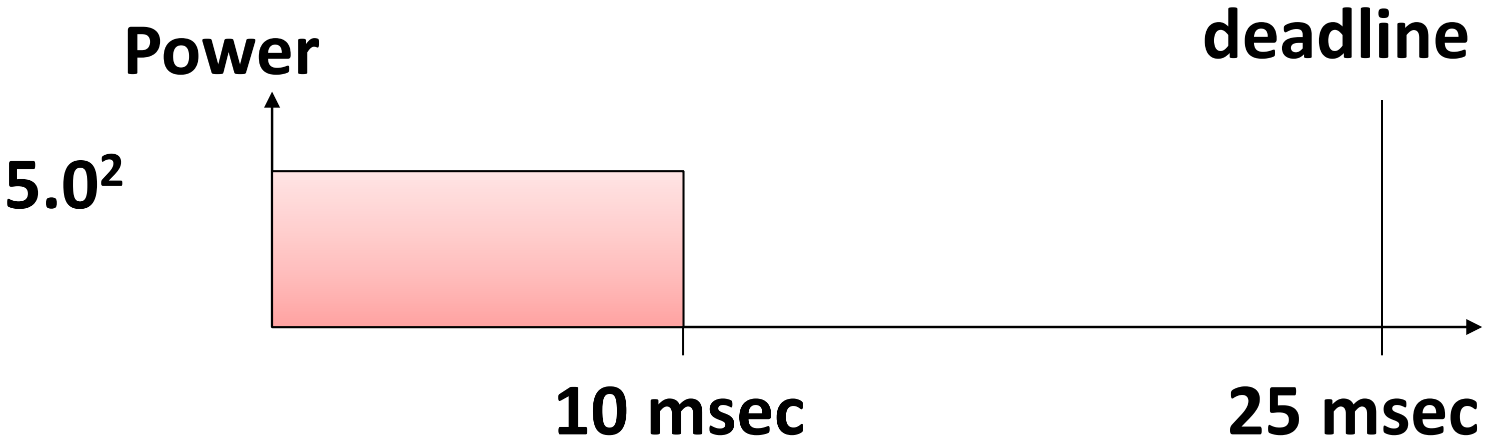
- Dynamic Voltage and Frequency Scaling
  - Intel SpeedStep
  - AMD PowerNow
- Started in laptops and mobile devices
- Now used in servers

DVS (Dynamic Voltage Scaling)

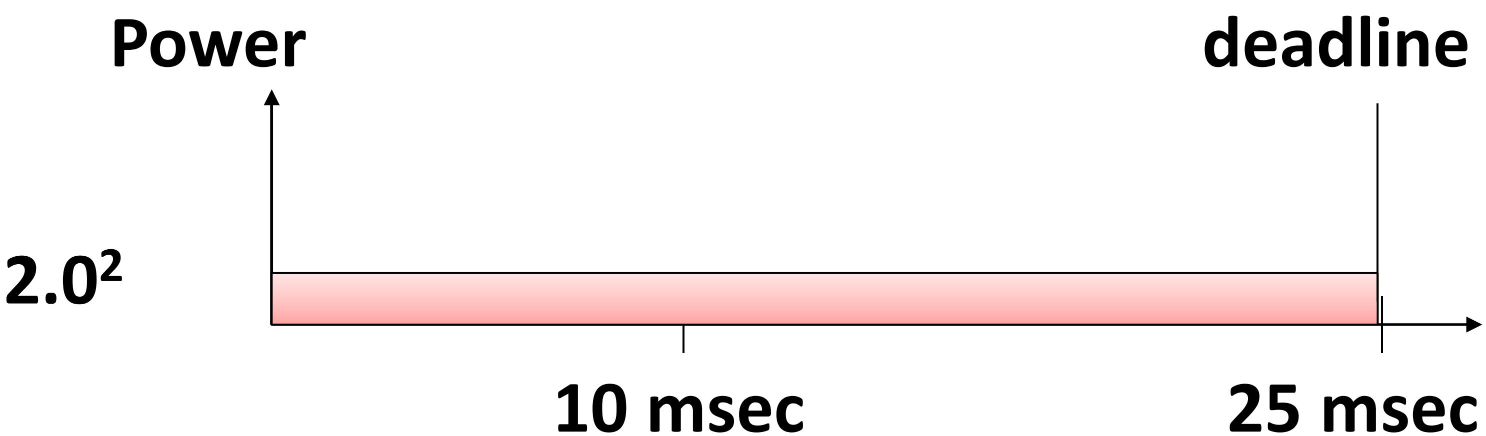
- Reducing the dynamic energy consumption
  - By lowering the supply voltage at the cost of performance degradation
- Recent processors support such ability
  - To adjust the supply voltage dynamically.
- The dynamic energy consumption
  - $\alpha * Vdd^2 * Ncycle$

    Vdd : the supply voltage, Ncycle : the number of clock cycle
  - **½ C V F^2 with V proportional to F ➔ α f^3**

DVS (Dynamic Voltage Scaling)



(a) Supply voltage = 5.0 V



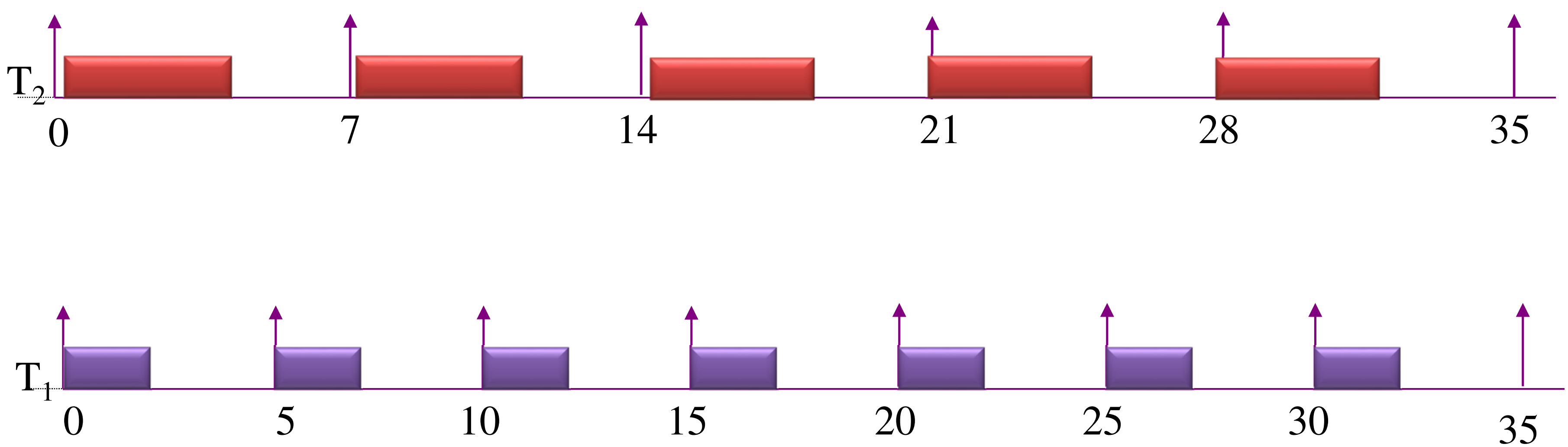
(b) Supply voltage = 2.0 V

Throttling Vs Over clocking

- Throttling
  - to hold somebody tightly by the throat and stop him/her breathing
  - Put a cut-off mark: Example car governor
  - Some thing going wrong: reduce activity
  - **Thermal/Power Throttling**
- Overclocking (If necessary): Turbo Boost
  - Put maximum doable afford
  - Run at maximum speed
  - Urgency to do more work

Periodic Task: Real Time Scheduler

- Task with periods
- Each task have to finish before deadline with in the period



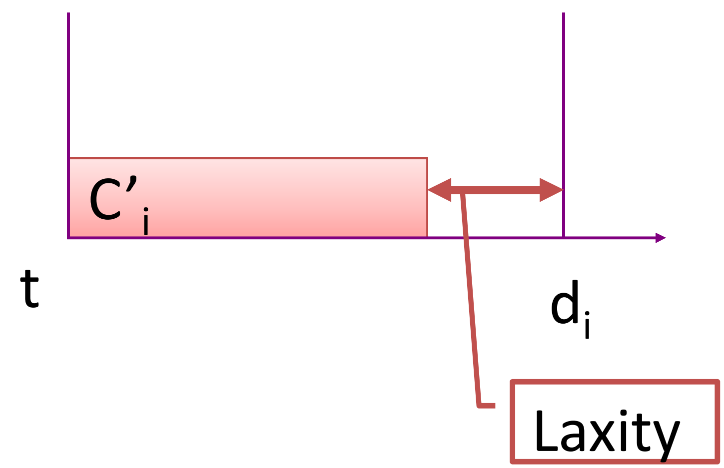
Periodic Tasks

- Necessary schedulability test
  - Sum of utilization factors μ_i must be less than or equal to n, where n is the number of processors
  - $\mu = \Sigma (c_i / p_i) <= n$
  - μ_i = Percentage of time the task T_i requires the service of a CPU

Periodic Task: Real Time Scheduler

Assumptions & Definitions

- Tasks are periodic
- No aperiodic or sporadic tasks
- Job (instance) deadline = end of period
- Tasks are preemptable



- Laxity of a Task

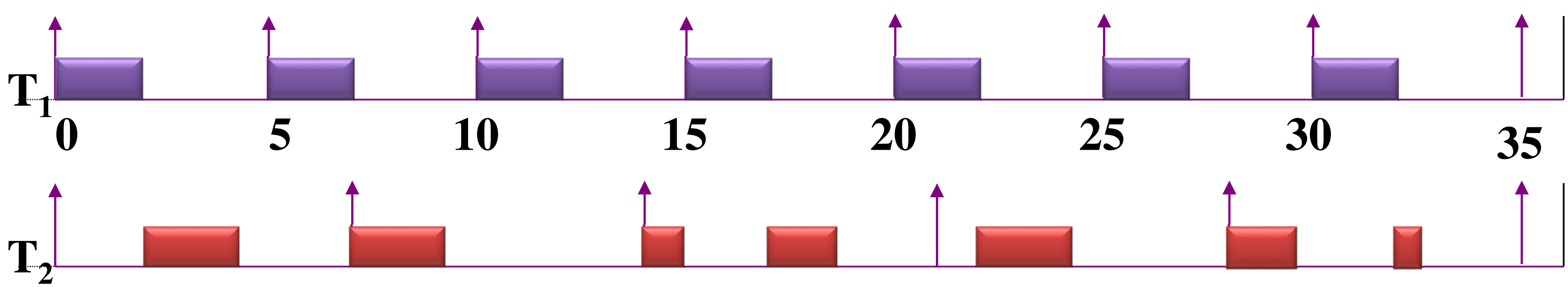
$$T_i = d_i - (t + c_i')$$

where di: deadline;

t : current time; c_i' : remaining computation time.

Rate Monotonic Scheduling

- **Static Scheduling**
  - Task with the smallest period is assigned the highest priority.
  - At any time, the highest priority task is executed.



Rate Monotonic (RM) Scheduling

- **Schedulability check (off-line)**

  \- A set of ***n* tasks** is schedulable on a uniprocessor by the RMS algorithm if the processor utilization (utilization test):

$$\boxed{\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{1/n} - 1)}$$

The term $n(2^{1/n} - 1)$ approaches $\ln 2$, (≈ 0.69 as $n \to \infty$).

Earliest Deadline First (EDF)

- **Dynamic Scheduling**
- Task with the smallest deadline/laxity is assigned the highest priority. EDF or <u>**Least Laxity First (LLF)**</u>
  - At any time, the highest priority task is executed.
- **Schedulability check (off-line)**
  - A set of <u>***n*** tasks</u> is schedulable on a uniprocessor by the EDF algorithm if the processor utilization.

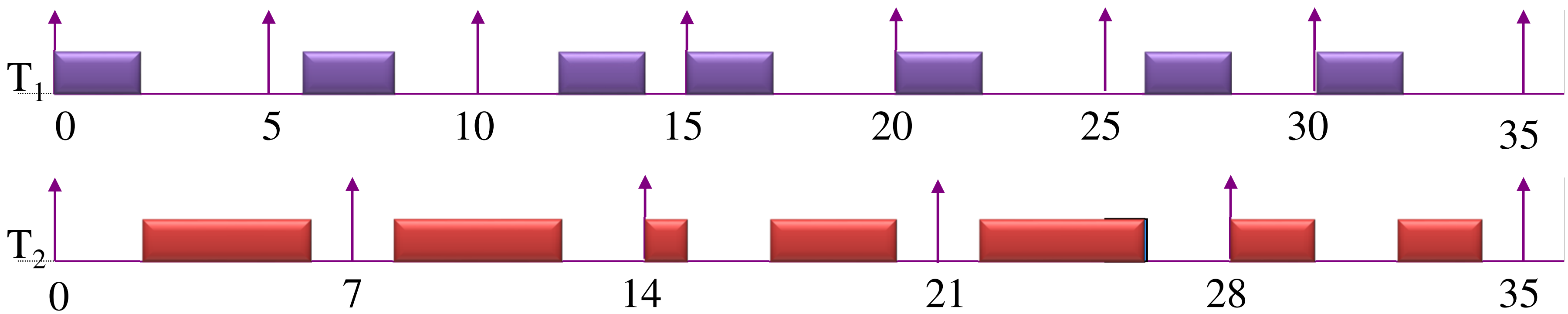
$$\sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$$

- This condition is both <u>necessary</u> and <u>sufficient.</u>

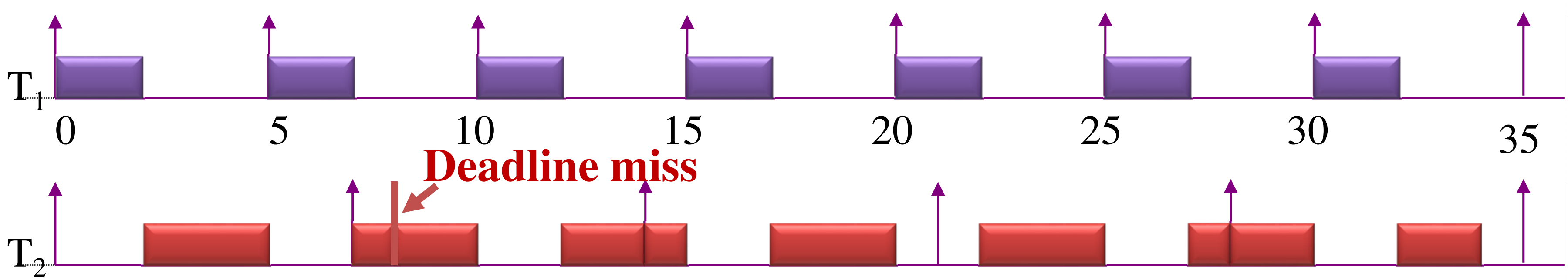
RM & EDF -- Example

Process	Period, T	WCET, C
T_1	5	2
T_2	7	4

EDF schedule



RMS schedule



Introduction to Scheduling Algorithms

A bit of Theoretical View

Overview

- Scheduling System Oriented
  - FCFS, SJF, Priority, RR
  - Multi Level Queue, MLQ with feedback
- Scheduling Algorithm
  - **Introduction to Scheduling Algorithms**

Scheduling Problems

- In a scheduling problem
  - One has to find time slots in which activities should be processed under given constraints.
- The main constraints are
  - Resource constraints and
  - Precedence constraints between activities
- A quite general scheduling problem is
  - ***Resource Constrained Project Scheduling Problem***
  - In short **RCPSP**

Parallel Machine Problems

- P: For **identical machines** $M_1, \ldots, M_m$
  - The processing time for ***j*** is the same on each machine.
- Q: For **uniform machine**
  - if $p_{jk} = p_j/r_k$.
- R: For **unrelated machines**
  - The processing time p_{jk} depends on the machine M_k on which j is processed.

Example: Machine Environment

	M1	M2	M3
P1	5	5	5
P2	8	8	8
P3	6	6	6

Identical

	M1	M2	M3
P1	5	4	6
P2	9	8	4
P3	3	18	4

Unrelated

	M1	M2	M3
P1	5	5/1.5	5/2
P2	9	9/1.5	9/2
P3	9	9/1.5	9/2

Uniform

Classification of Scheduling Problems

Classes of scheduling problems can be specified in terms of the three-field classification

$$\alpha \mid \beta \mid \gamma$$

where

- α specifies the **machine environment**,
- β specifies the **job characteristics**, and
- γ describes the **objective function(s).**

Machine Environment

- 1 single machine
- P parallel identical machines
- Q uniform machines
- R unrelated machines
- MPM multipurpose machines, J job-shop,
- F flow-shop O open-shop

If the number of machines is fixed to m we write **Pm, Qm, Rm**, MPMm, Jm, Fm, Om.

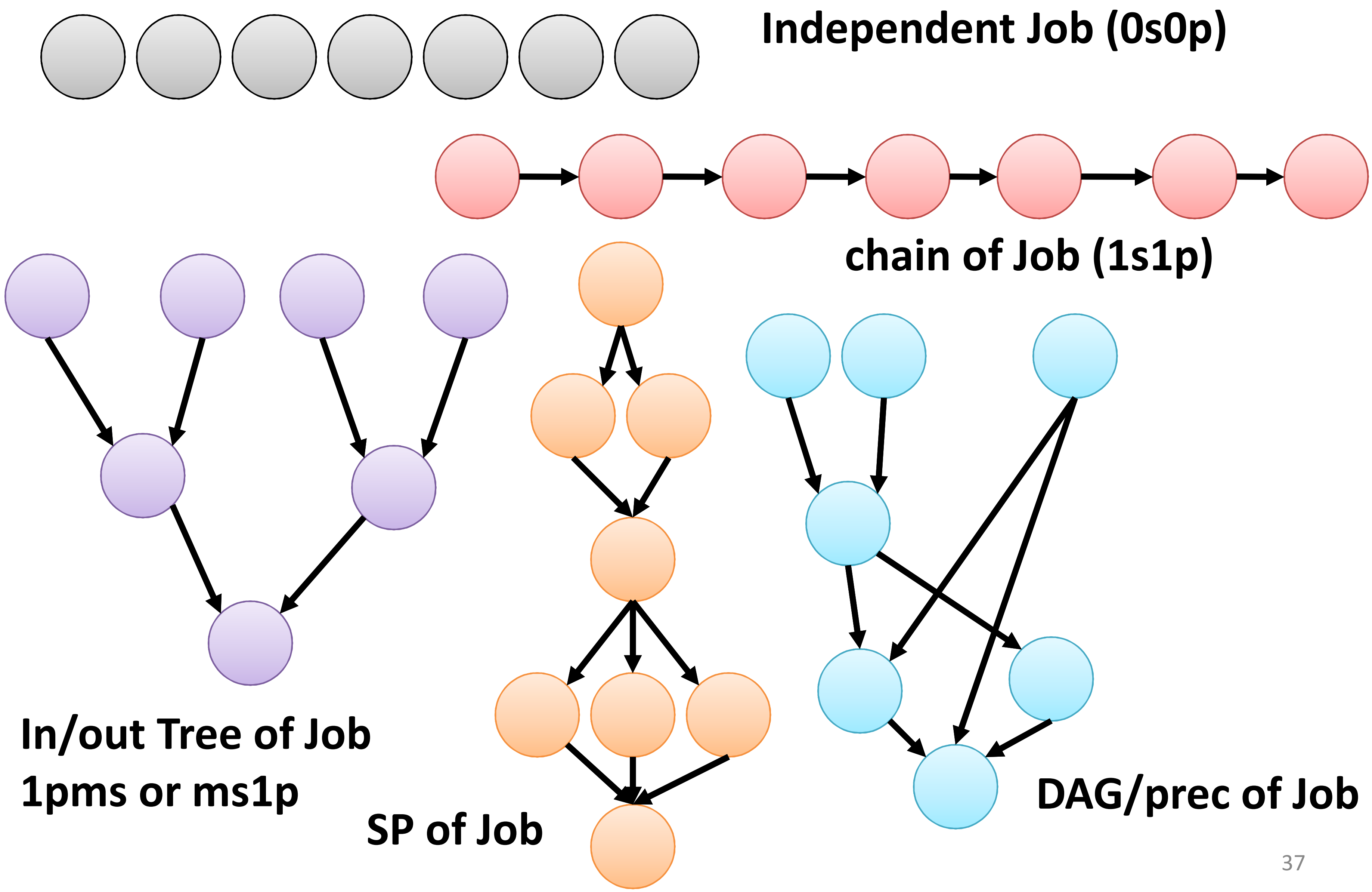
Job Characteristics

- **pmtn** preemption
- r_j release times /arrival time
- d_j deadlines
- $p_j = 1$ **or** $p_j = p$ **or** $p_j \in \{1,2\}$
  restricted processing times

Job Characteristics

- **prec** arbitrary precedence constraints
- **intree (outtree)** intree (or outtree) precedences
- **chains** chain precedences
- **series-parallel** a series-parallel precedence graph

Job Precedence Examples



Objective Functions

Two types of objective functions are most common:

- **bottleneck objective functions** $\max \{f_j(C_j) \mid j = 1, \ldots, n\}$, and
- **sum objective functions** $\sum f_j(C_j) = f_1(C_1) + f_2(C_2) + \ldots \ldots + f_n(C_n)$.

Objective Functions

- $\mathbf{C_{max}}$ and $\mathbf{L_{max}}$ symbolize the bottleneck objective functions with
  - $f_j(C_j) = C_j$ (makespan)
  - $f_j(C_j) = C_j - d_j$ (maximum lateness)
- Common sum objective functions are:
  - $\Sigma\, C_j$ (mean flow-time)
  - $\Sigma\, \omega_j C_j$ (weighted flow-time)

Objective Functions

- **Number of Late Job**
  - ΣU_j (number of late jobs) and $\Sigma \omega_j U_j$ (weighted number of late jobs) where $U_j = 1$ if $C_j > d_j$ and $U_j = 0$ otherwise.
- **Tardiness**
  - ΣT_j (sum of tardiness) and $\Sigma \omega_j T_j$ (weighted sum of tardiness)
  - Tardiness of job j is given by

$$T_j = \max \{ 0, C_j - d_j \}.$$

Examples

- $1 \mid prec; p_j = 1 \mid \Sigma\, \omega_j C_j$
- $P2 \mid \mid C_{max}$
- $P \mid p_j = 1; r_j \mid \Sigma\, \omega_j U_j$
- $R2 \mid chains; pmtn \mid C_{max}$
- $P3 \mid n = 3 \mid C_{max}$
- $Pm \mid p_{ij} = 1; outtree; r_j \mid \Sigma\, C_j$

Example: $1|C_{max}$

- N independent job without pre-emption
- 1 processor
- Minimize C_{max}
- Sol: Schedule in any orders

Example: $1|\sum C_i$

- N independent job without pre-emption
- 1 processor
- Minimize $\sum C_i$
- Sol: Schedule shortest processing time first
  - SJF is optimal

Example: $1|\sum w_i C_i$

- N independent job without pre-emption
- 1 processor
- Minimize $\sum w_i C_i$
- Sol:
  - Calculate processing time to weight ratio
  - Rank jobs in increasing order of p_i/w_i and schedule accordingly
  - The Weighted Shortest Processing Time First rule is Optimal for $\mathbf{1|\sum w_i C_i}$

Example: $1|chain|\sum w_iC_i$

- N independent jobs with chain precedence without pre-emption
- 1 processor, multiple chain
- Minimize $\sum w_iC_i$
- Sol:
  - Calculate processing time to weight ratio (ρ) of chains (by including a number of tasks from a chains)
  - Process the tasks from chain till the ρ of the chain is higher than others chain

Example: $1|prec|\sum w_i C_i$

- For general precedence the problem is Hard
- NP-Complete problem

Thanks