CS343: Operating System

Process Scheduling

Lect11 : 21st Aug 2023

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Popular Scheduling

- FCFS, SJF, SJF-I, Priority, Priority-I
- Round-Robin Scheduler
- Multi-Level Priority Queue
  - Feed Back Priority Queue
- Real time Scheduler
- Energy Efficient with DPM and DVFS

Round Robin (RR)

- **Gol Gappe Wala Scheduling**
- Each process gets a small unit of CPU time (**time quantum *q***)
  - Usually 10-100 milliseconds.
  - After this time has elapsed, the process is preempted and added to the end of the ready queue.

Round Robin (RR)

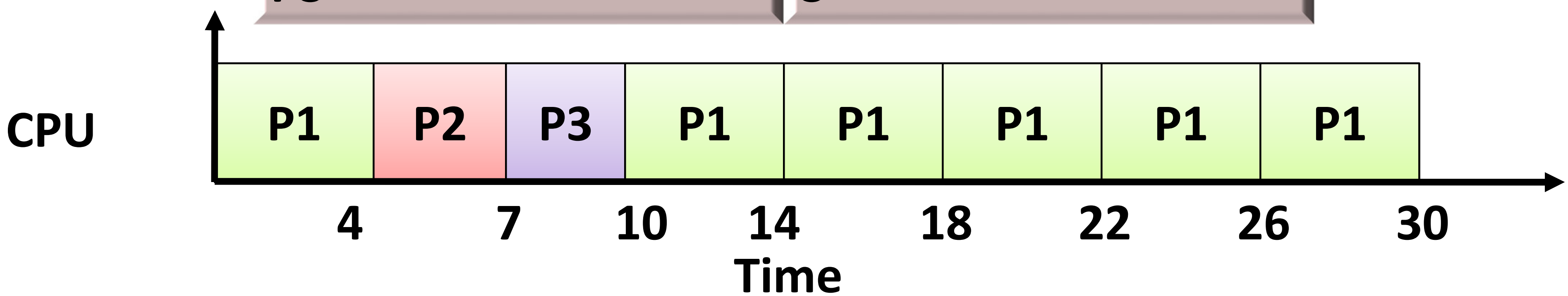
- If there are ***n*** processes in the ready queue and the time quantum is ***q***, then
  - Each process gets **1/*n*** of the CPU time in chunks of at most ***q*** time units at once.
- No process waits more than **(*n*-1)*q*** time units.

Round Robin (RR)

- Timer interrupts every quantum to schedule next process
  - **Timer: Hardware unit, similar to Alarm**
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

Example : RR (with q = 4)

Process	Burst Time
P1	24
P2	3
P3	3



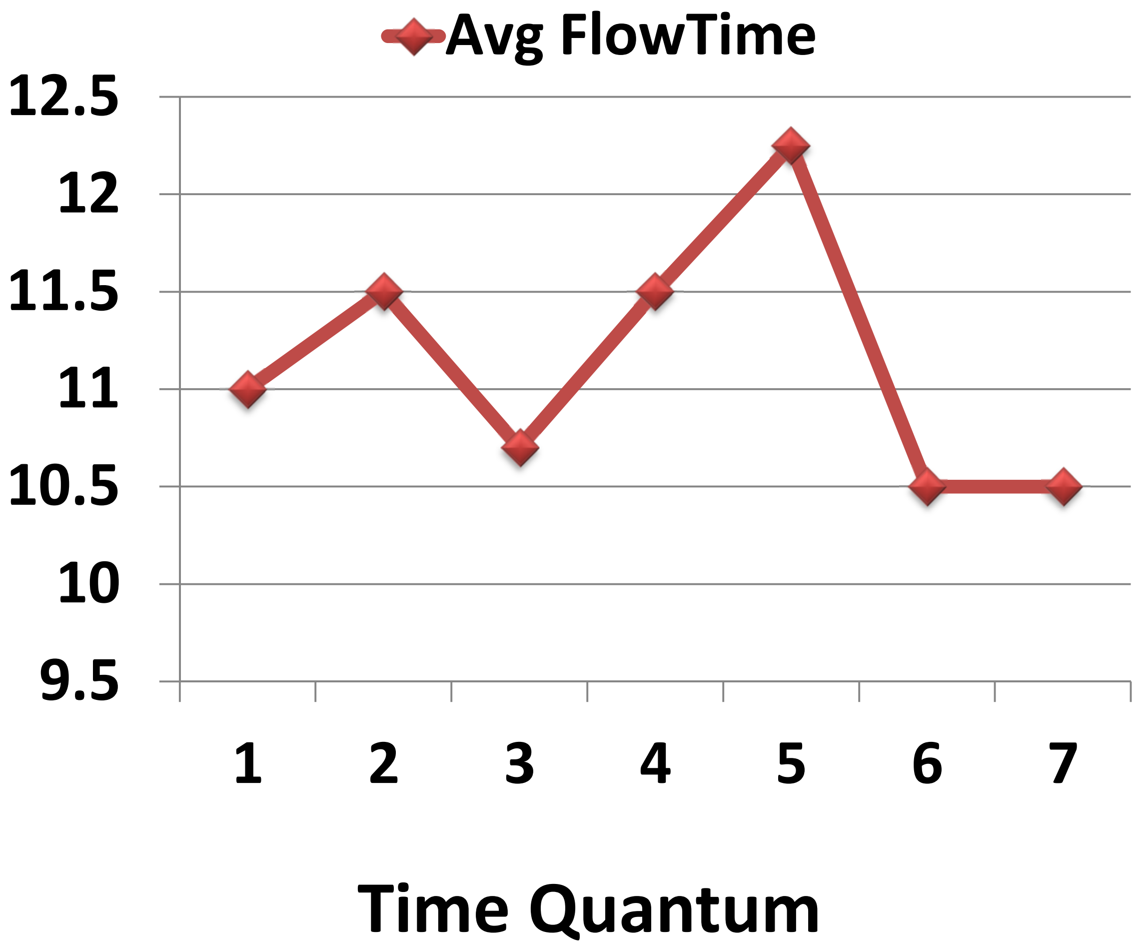
- Typically, higher Average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

Time Quantum and Context Switch Time

Process time 10

Quantum	Context Switches
12	0
6	1
1	9

Turnaround Time Varies With The Time Quantum



Process	Time
P1	6
P2	3
P3	1
P4	7

80% of CPU bursts should be shorter than q

Multilevel Queue

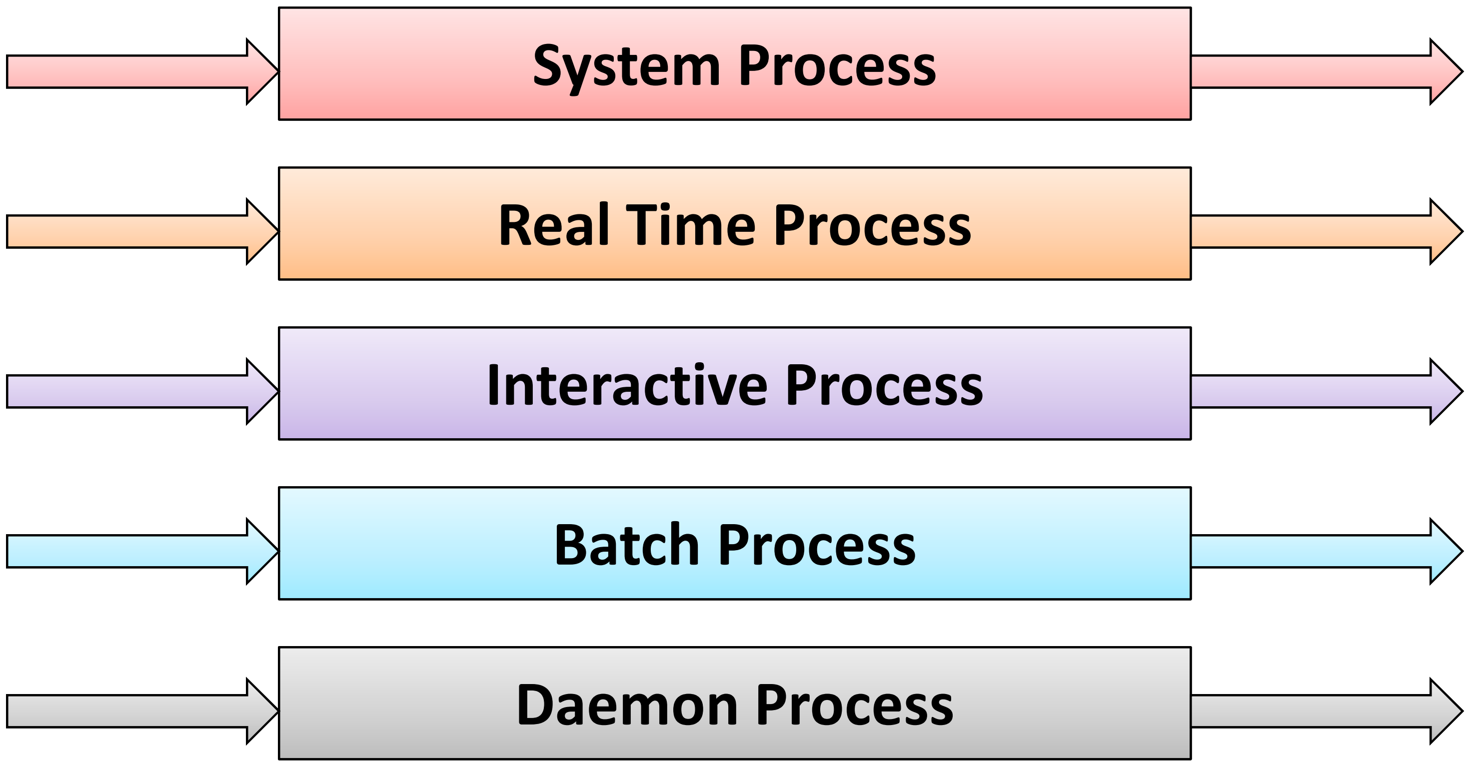
- Ready queue is partitioned into separate queues, eg:
  - **foreground** (interactive)
  - **background** (batch)
- Process permanently in a given queue
- Each queue has its own scheduling algorithm:
  - foreground – RR
  - background – FCFS

Multilevel Queue

- Scheduling must be done between the queues:
  - Fixed priority scheduling; (i.e., serve all from foreground then from background). **Possibility of starvation.**
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes;
  - i.e., 80% to foreground in RR, 20% to background in FCFS

Multilevel Queue Scheduling

Highest Priority



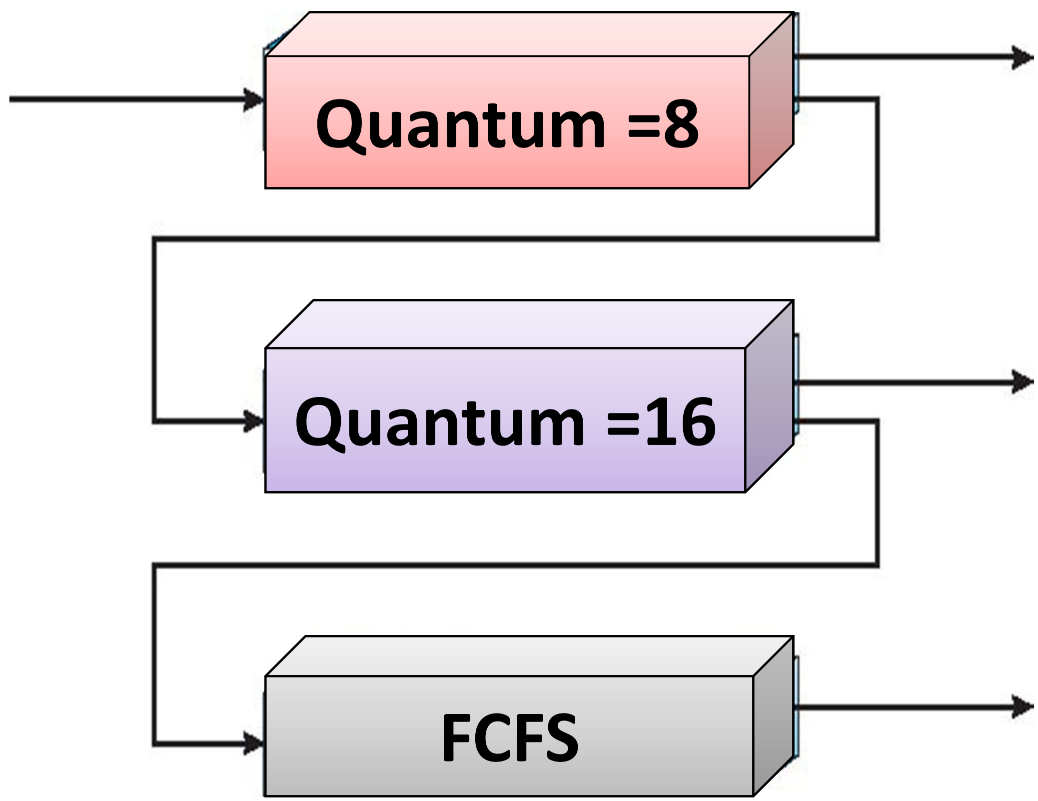
Lowest Priority

Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

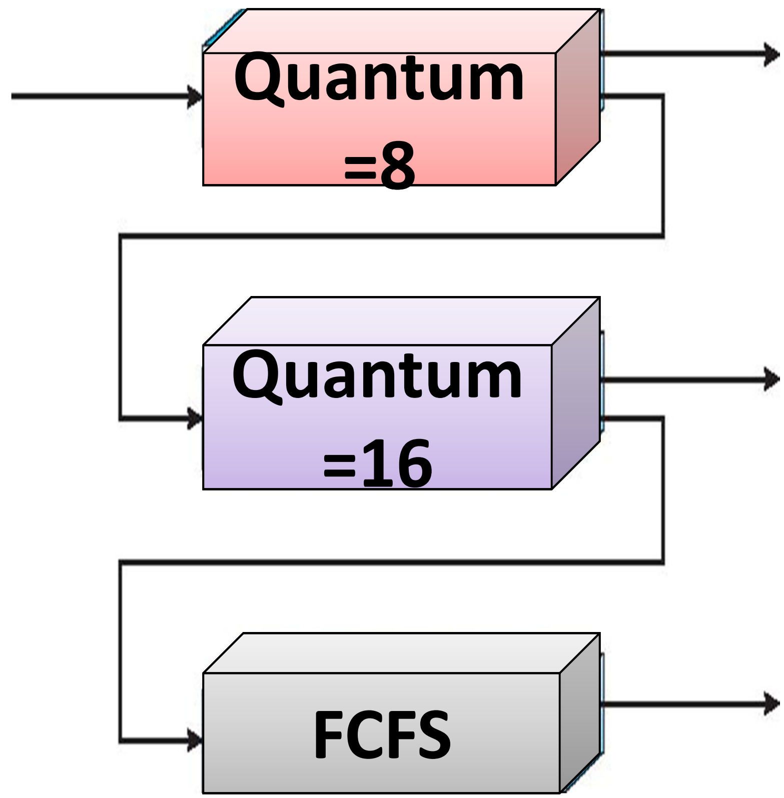
Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS



Example of Multilevel Feedback Queue

- A new job enters queue Q_0 which is served FCFS
  - When it gains CPU, job receives 8 milliseconds
  - If it does not finish in 8 milliseconds, job is moved to queue Q_1
- At Q_1 job is again served FCFS and receives 16 additional milliseconds
  - If it still does not complete, it is preempted and moved to queue Q_2



Scheduling: When No Scarcity of CPU Resource

Multicore System

- CPU can run at different frequencies
  - DVFS : P=Ps+ f^3
  - **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? Yes

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

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.

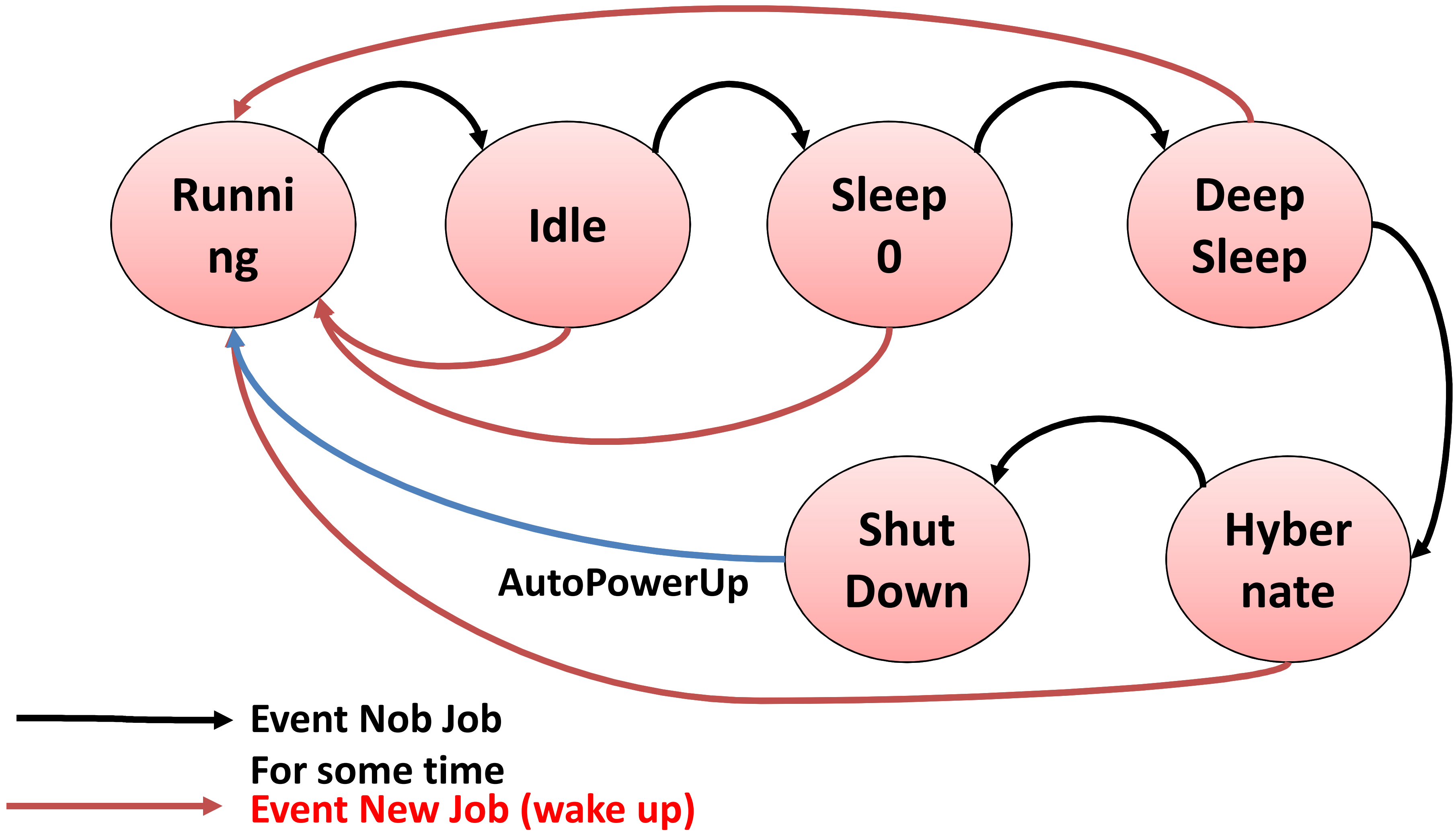
DVFS-based Power Aware Scheduling : Motivation

- 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

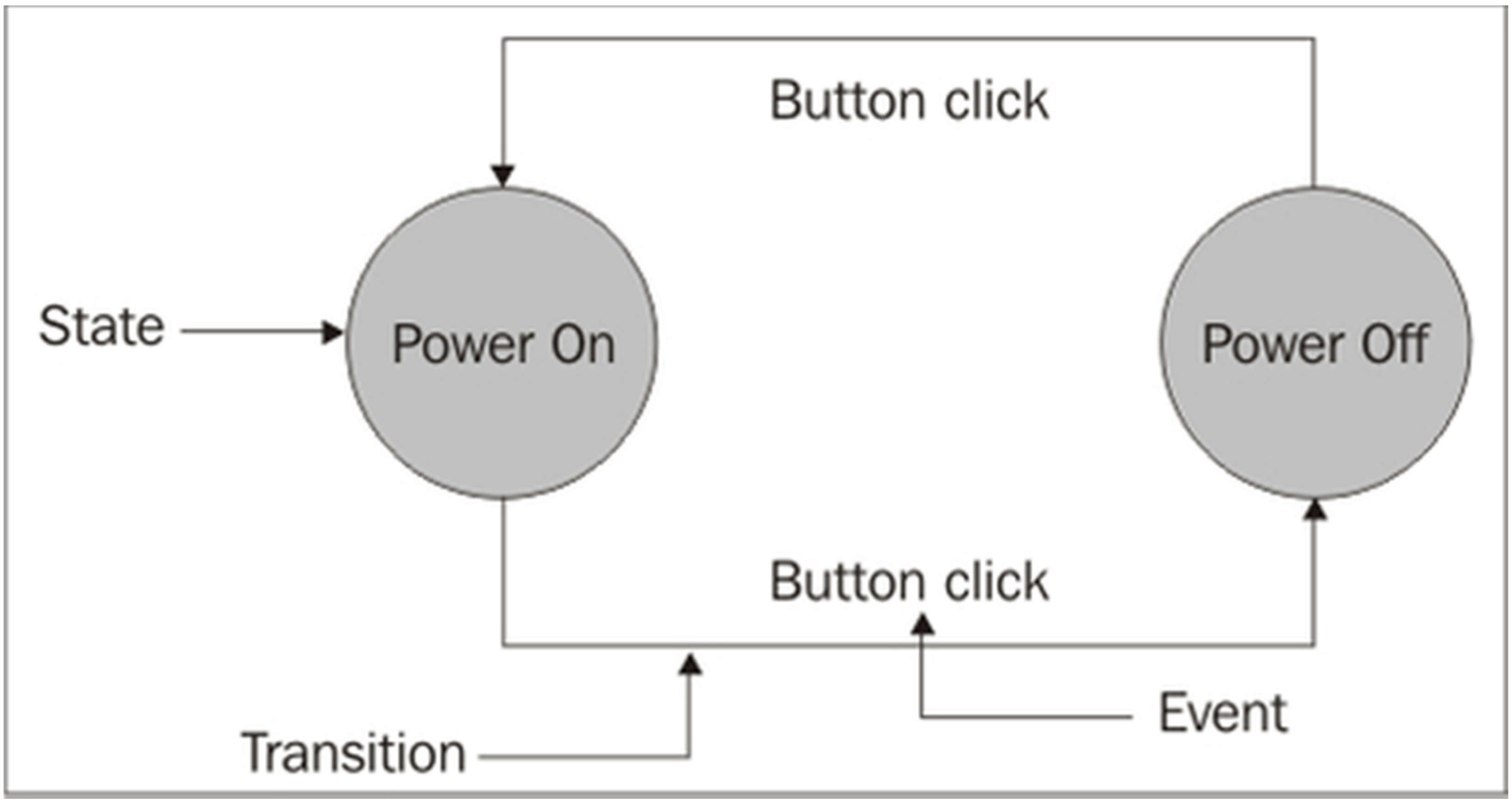
Static PM vs Dynamic PM

- 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**

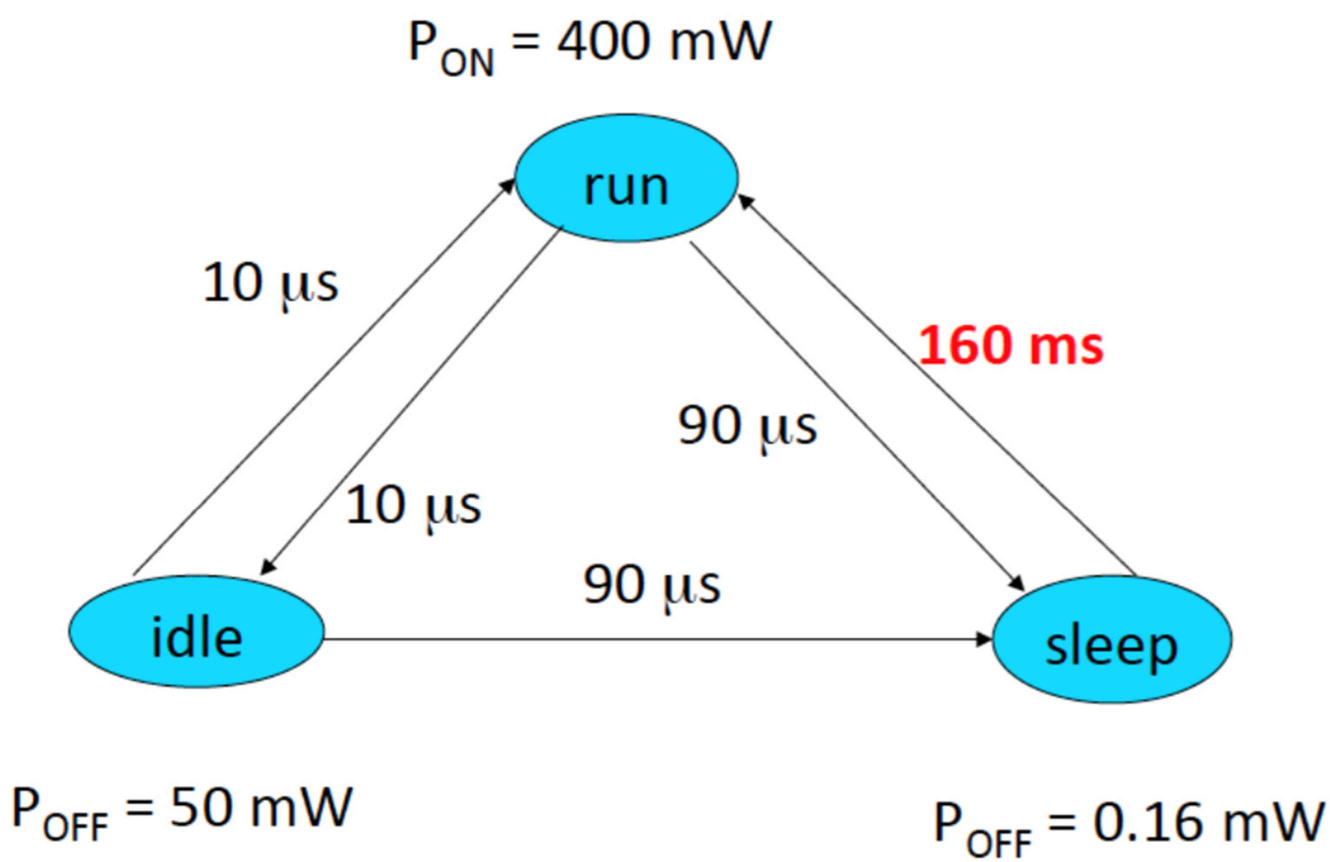
Static PM with Power States



Static PM with Power States

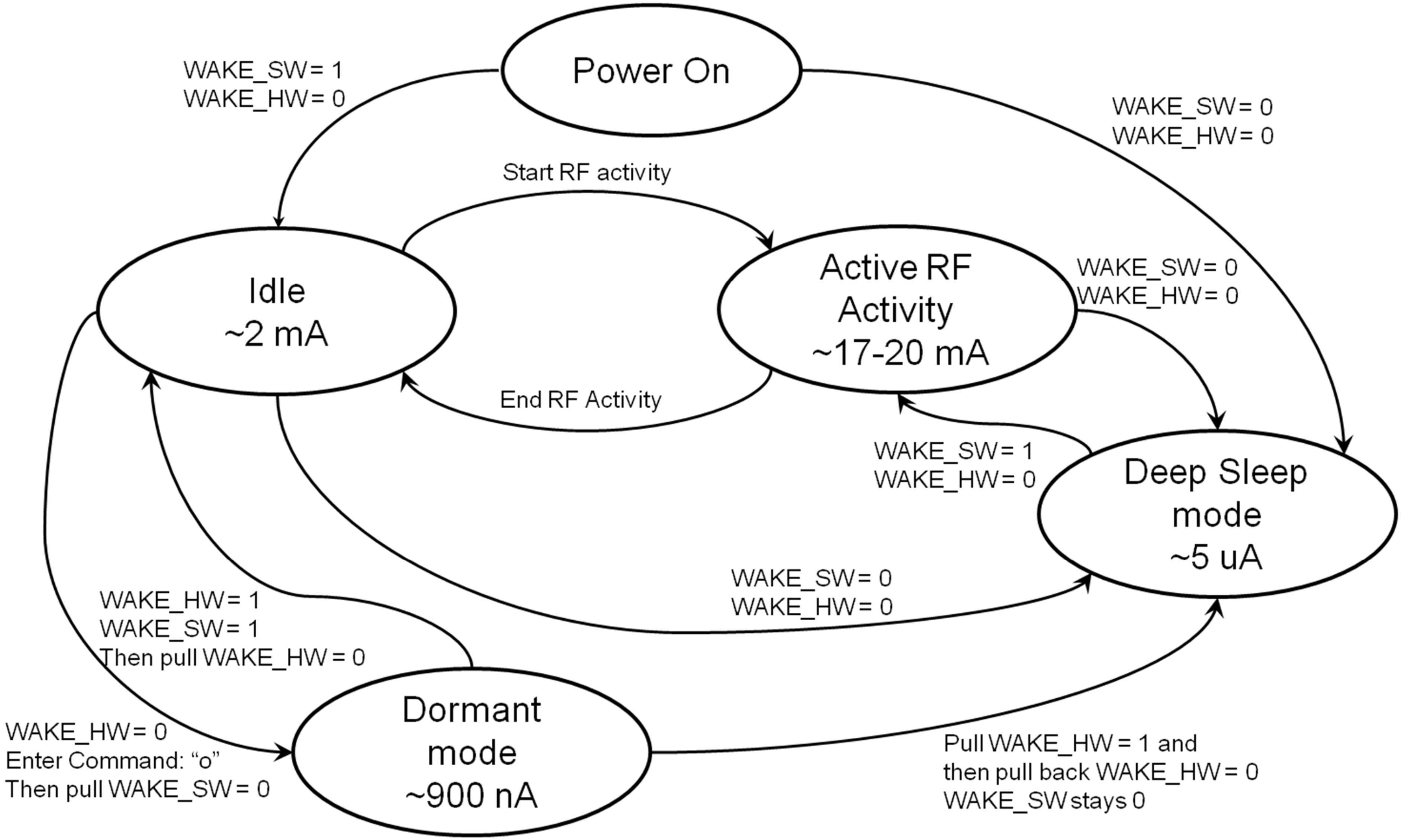


Static PM with Power States



$P_{TR} = P_{ON}$

Static PM with Power States



Static PM vs Dynamic PM

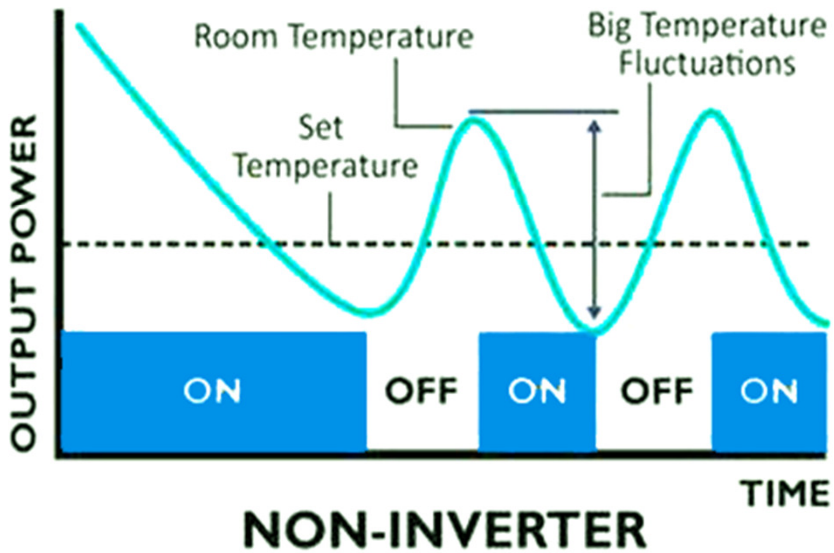
- Static PM
  - Invokes by user does not depends on user activities
  - Power down mode: c0, c1, …cm
    - Off, dose, nap, sleep, run
  - Mode exit upon receiving an interrupt
  - Power State machine
- Dynamic PM
  - Control power based on dynamic activity in CPU
  - Dynamically change freq, shut some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

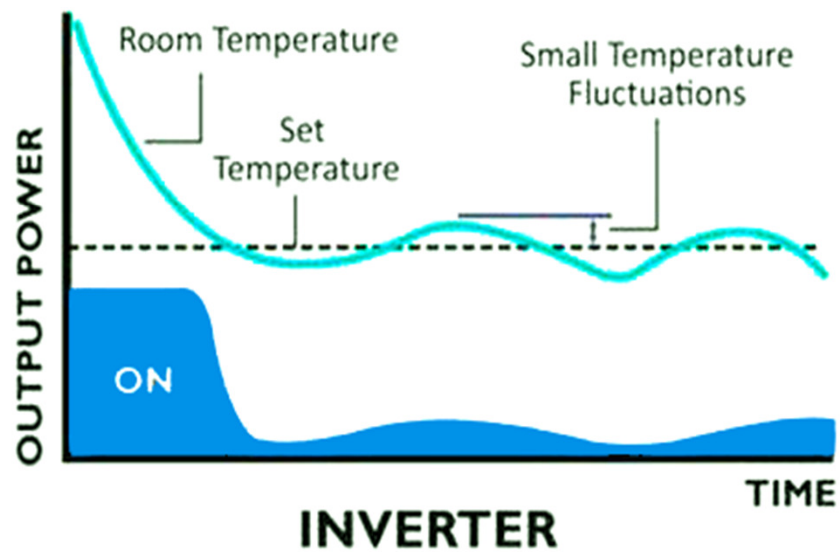
Real Life Issue: Inverter AC

- 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





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

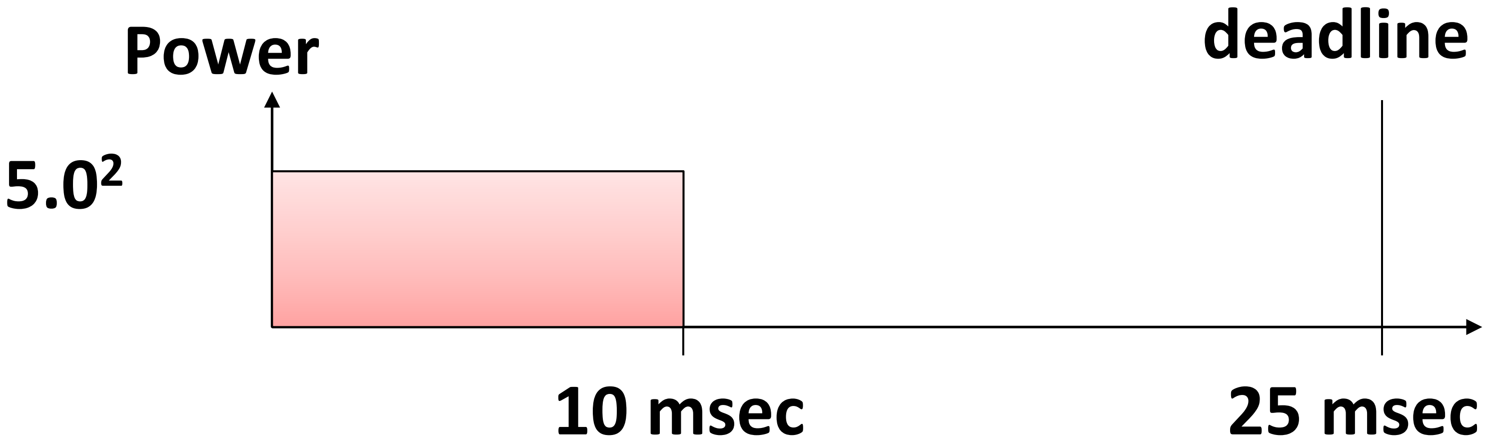
- Dynamic Voltage and Frequency Scaling
  - Intel SpeedStep
  - AMD PowerNow
- Started in laptops and mobile devices
- Now used in servers

DVS (Dynamic Voltage Scaling)

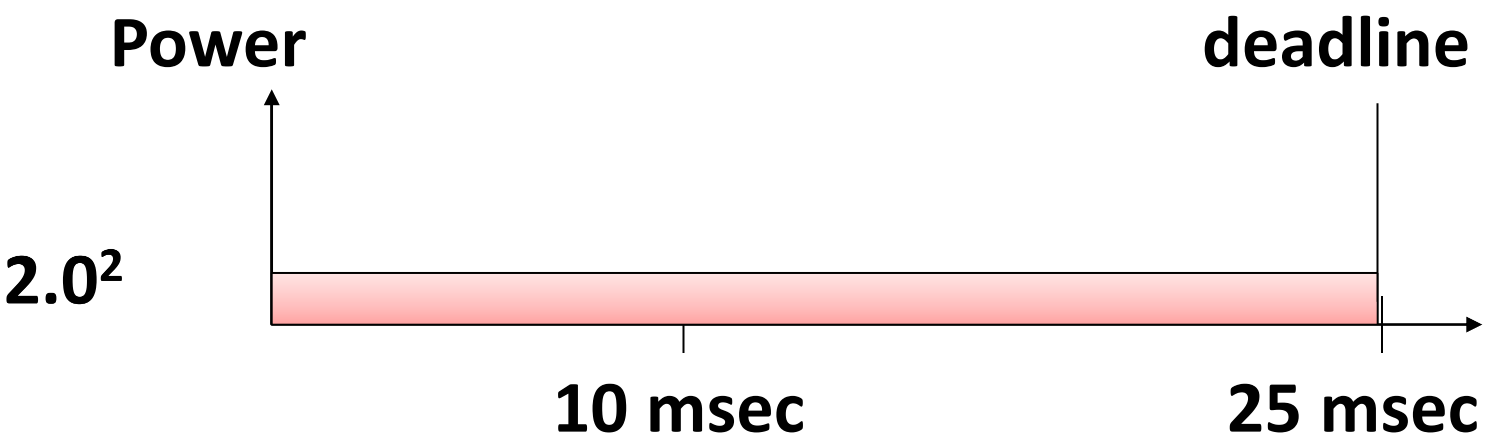
- Reducing the dynamic energy consumption
  - By lowering the supply voltage at the cost of performance degradation
- Recent processors support such ability
  - To adjust the supply voltage dynamically.
- The dynamic energy consumption
  - $\alpha * Vdd^2 * Ncycle$

    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

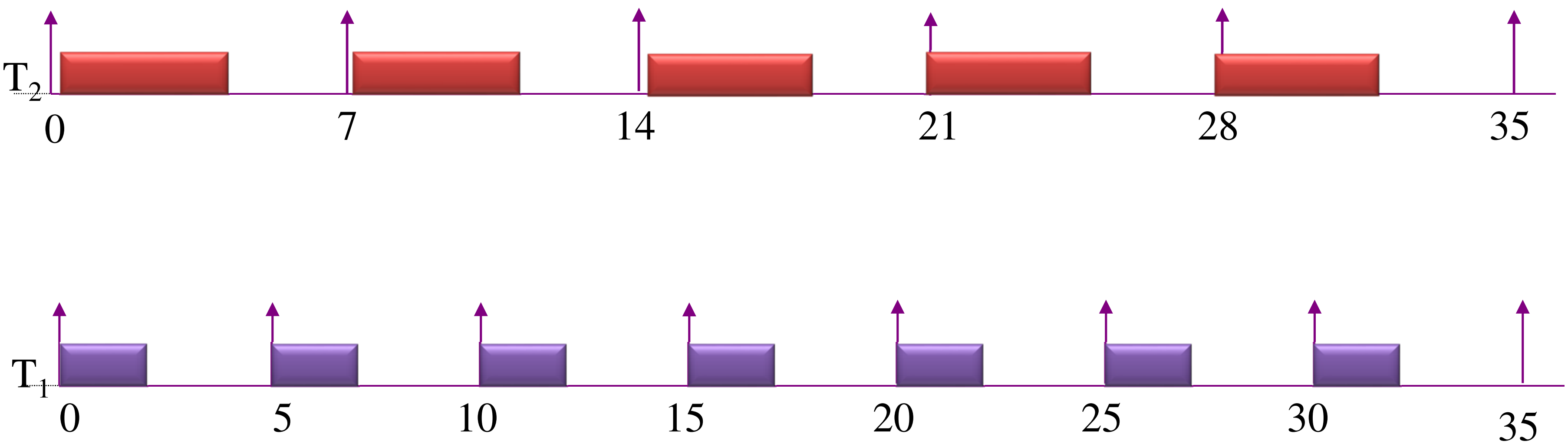
Real Time Scheduling

Real Time Scheduling

- MPEG, Audio
  - 30 frame/Sec
- Can you run UHD video file on Mobile ?
- Periodic Task
- **Nice Value** in **Linux**
  - 0-100 for real time task, 101-140 non real time task
  - Size of processor quantum (share) based on nice value

Periodic Task: Real Time Scheduler

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



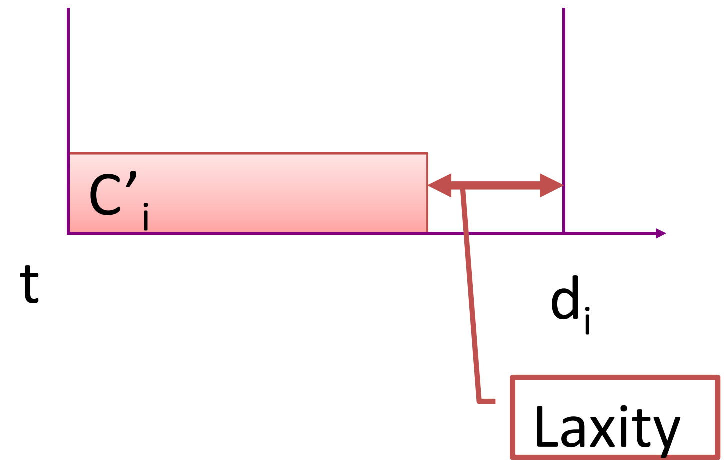
Periodic Tasks

- Necessary schedulability test
  - Sum of utilization factors μ_i must be less than or equal to n, where n is the number of processors
  - $\mu = \sum (c_i / p_i) <= n$
  - 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

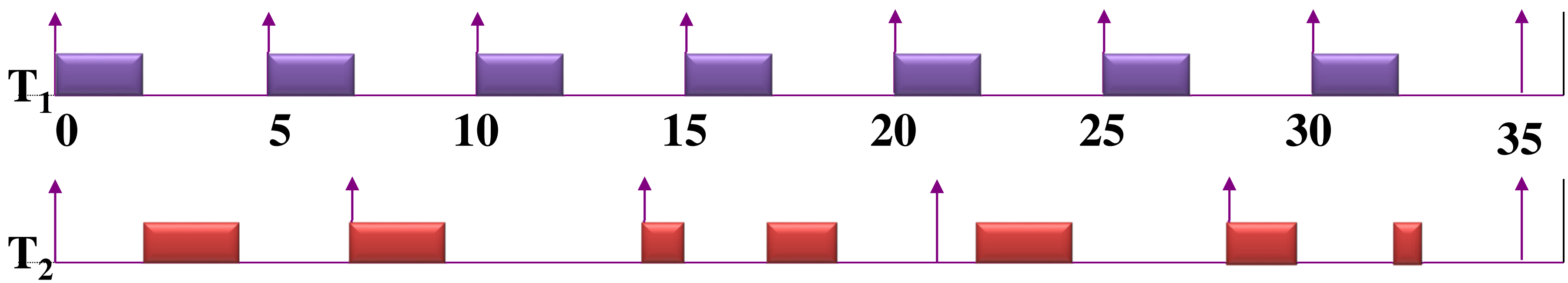
$$T_i = d_i - (t + c_i')$$

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):

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

The term $n(2^{1/n} -1)$ approaches $\ln 2$, ($\approx$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 **Least Laxity First (LLF)**
  - At any time, the highest priority task is executed.
- **Schedulability check (off-line)**
  - A set of ***n* tasks** is schedulable on a uniprocessor by the EDF algorithm if the processor utilization.

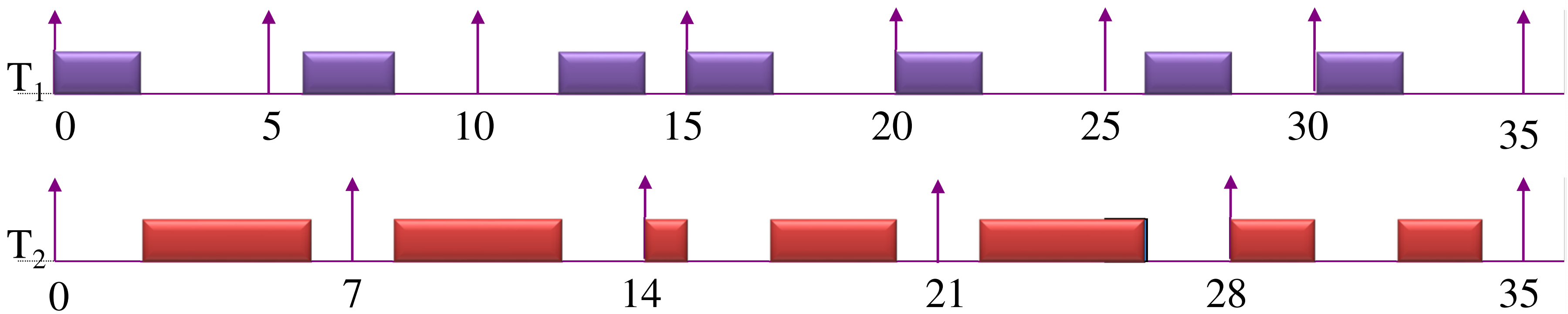
$$\boxed{\sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1}$$

- This condition is both necessary and sufficient.

RM & EDF -- Example

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

EDF schedule



RMS schedule