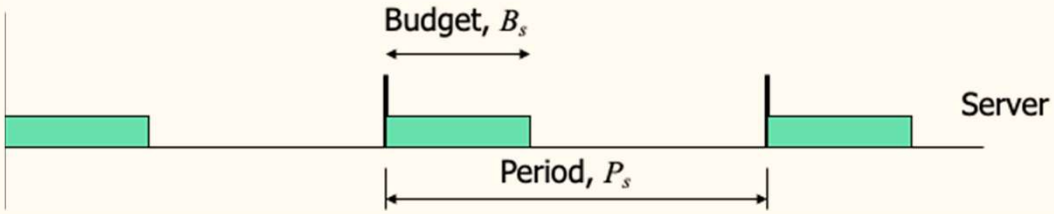
Midterm Review

Mixed Periodic and Aperiodic Task Systems

- Idea: aperiodic tasks can be served by periodically invoked servers
- The server can be accounted for in periodic task schedulability analysis
- The server has a period P_s and a budget B_s
- Server can serve aperiodic tasks until budget expires
- Servers have different flavors depending on the details of when they are invoked, what priority they have, and how budgets are replenished



Things to Remember

Polling server: worst latency for aperiodic task

Deferrable servers: good latency for aperiodic tasks but worse schedulability for periodic tasks

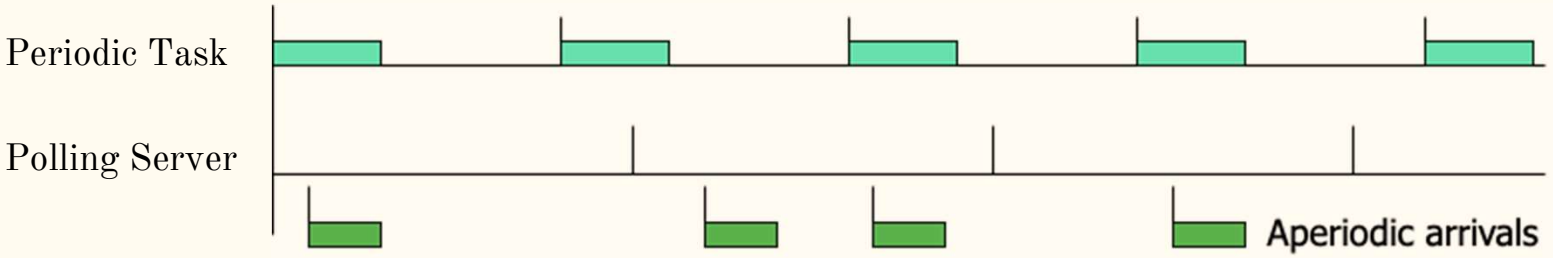
Sporadic and priority exchange servers: A compromise.

Polling Server

- Runs as a periodic task (priority set according to RM)
- Aperiodic arrivals are queued until the server task is invoked
- When the server is invoked it serves the queue until it is empty or until the budget expires then suspends itself
- If the queue is empty when the server is invoked it suspends itself immediately.
- Server is treated as a regular periodic task in schedulability analysis

Polling Server - Example

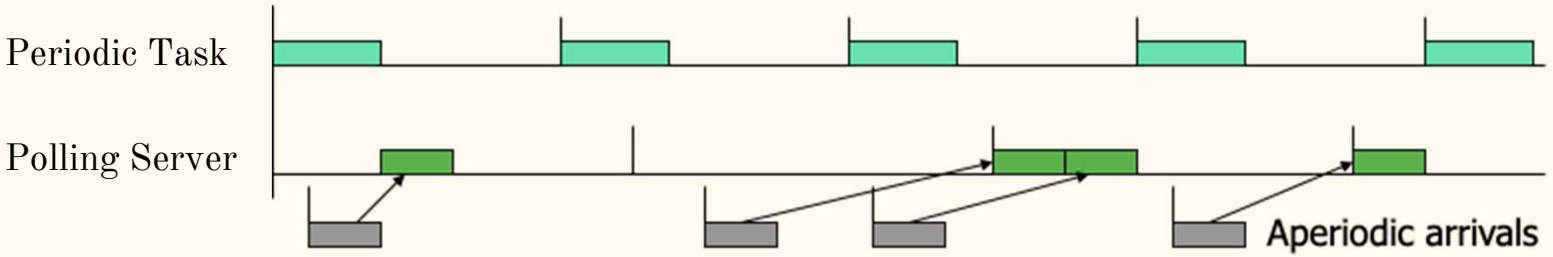
- Polling server:
  - Period Ps = 5
  - Budget Bs = 2
- Periodic task
  - P = 4
  - C = 1.5
- All aperiodic arrivals have C=1



Polling Server - Example

- Polling server:
  - Period Ps = 5
  - Budget Bs = 2
- Periodic task
  - P = 4
  - C = 1.5
- All aperiodic arrivals have C=1

When it is time to run the server, if there are no aperiodic tasks to serve, it goes to sleep until next period



Deferrable Server

- Keeps the balance of the budget until the end of the period

- Executes aperiodic tasks immediately at any time as long as the budget is not depleted.

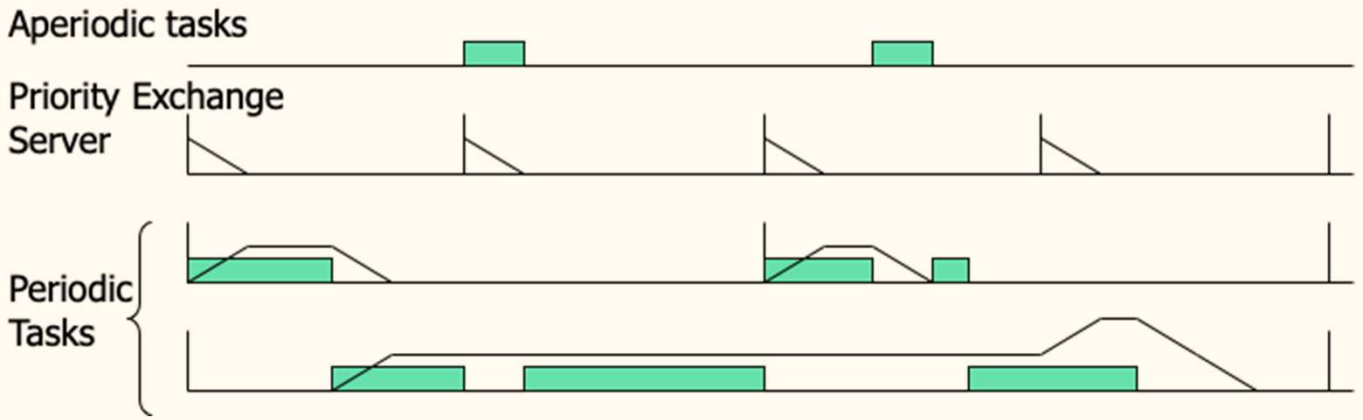
Deferrable Server

- Keeps the balance of the budget until the end of the period
- Executes aperiodic tasks immediately at any time as long as the budget is not depleted.
- Problem: Can result in a more "bursty" interference with other tasks (e.g., wake up right before end of its period to execute an aperiodic task then continue at the beginning of the next period to execute another)

Priority Exchange Server

- Like the deferrable server, it keeps the budget until the end of server period

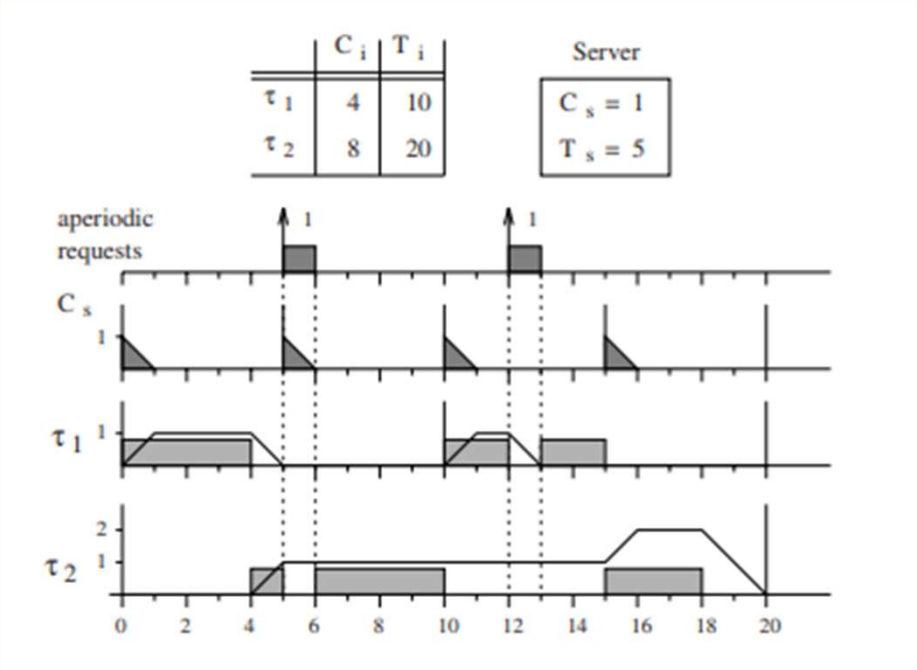
- Unlike the deferrable server the priority slips over time: When not used the priority is exchanged for that of the executing periodic task.



Priority Exchange Server - How it works ?

- When a priority exchange occurs between a periodic task and a PE server, the periodic task executes at the priority level of the server while the server accumulates a capacity at the priority level of the periodic task.
- The periodic task advances its execution, and the server capacity is not lost but preserved at a lower priority.
- If no aperiodic requests arrive to use the capacity, priority exchange continues with other lower-priority tasks until either the capacity is used for aperiodic service or it is degraded to the priority level of background processing.
- All priority ties are broken in favor of aperiodic tasks.

Priority Exchange Server - Example



From Hard Real Time Book. Example 5.14

Sporadic Server

- Server is said to be active if it is in the running or ready queue, otherwise it is idle.
- When an aperiodic task comes and the budget is not zero, the server becomes active

Sporadic Server

- Server is said to be active if it is in the running or ready queue, otherwise it is idle.
- When an aperiodic task comes and the budget is not zero, the server becomes active
- Every time the server becomes active, say at t_A, it sets replenishment time one period into the future, $t_A + P_s$ (but does not decide on replenishment amount).
- When the server becomes idle, say at t_I , set replenishment amount to capacity consumed in $[t_A, t_I]$

Multicore Scheduling

- Partitioned

- Global

Multicore Scheduling

- Partitioned
  - Each core has statically assigned tasks
  - Better isolation
  - Less effective load sharing (idle time on one core cannot be utilized by another

- Global

Multicore Scheduling

- Partitioned
  - Each core has statically assigned tasks
  - Better isolation
  - Less effective load sharing (idle time on one core cannot be utilized by another

- Global
  - A single queue of tasks is dispatched to whatever core is available
  - Better load sharing
  - Poor isolation

Multicore System Utilization

- Utilization, expressed below, for a system of m cores can be 0 to m:

$$U = \Sigma_i C_i / P_i$$

Utilization Bound for Partitioned EDF

- For a uniprocessor, a set of independent periodic tasks (with periods equal to deadline) is schedulable if $U \leq 1$.
- What about a partitioned multiprocessor?

  Schedulable by partitioned EDF if

  $$U \leq (m+1)/2 \qquad \text{(sufficient condition)}$$

Utilization Bound for Partitioned EDF

- For a uniprocessor, a set of independent periodic tasks (with periods equal to deadline) is schedulable if $U \leq 1$.
- What about a partitioned multiprocessor?

  Schedulable by partitioned EDF if

  $$U \leq (m+1)/2 \qquad \text{(sufficient condition)}$$

- There cannot be a better bound. Why?
- Consider m tasks of utilization (0.5 + very small value) that arrive first, then one more task of utilization = 0.5. The last task cannot be scheduled.

Utilization Bound for Partitioned EDF

- What if the largest-utilization task (also called the heaviest task) has a utilization no more than U_{max}?

- Task is schedulable if and only if

$$U \leq (\beta\, m + 1) / (m + 1)\,; \quad \beta = \text{floor}\,(1 / \text{Umax})$$

Utilization Bound for Global EDF

- Consider a case where m very small tasks arrive (each of nearly zero utilization), then a task of utilization = 1 arrives. Can the last task be scheduled?

Utilization Bound for Global EDF

- Consider a case where m very small tasks arrive (each of nearly zero utilization), then a task arrives of utilization = 1. Can the last task be scheduled?

- No

Utilization Bound for Global EDF

- What if maximum task utilization is U_{max}?

- The task set is schedulable if and only if $U \leq m - (m - 1) U_{max}$

Power of Computation

Terminology

- R : Power spent on computation
- V : Processor voltage
- f : Processor clock frequency
- R0 : Leakage power

Power spent on computation is:

- $R = k_v V^2 f + R_0$

  where k_v is a constant

Energy of Computation

- Power spent on computation is:

$$R = k_v V^2 f + R_0$$

- Consider a task of length C clock cycles and a processor operating at frequency f
- The execution time is $t = C/f$
- Energy spent is:
  - $E = R\,t = (k_v V^2 f + R_0)(C/f)$

Reducing Processor Frequency

- Does it make sense to operate the processor at a reduced speed to save energy? Why or why not?

Reducing Processor Frequency

- Does it make sense to operate the processor at a reduced speed to save energy? Why or why not?
- Possible Answer:
  - $E = R\,t = (k_v V^2 f + R_0)(C/f) = k_v V^2 C + R_0 C/f$
- Conclusion: E is minimum when f is maximum.
  - Operate at top speed

Reducing Processor Frequency

- Does it make sense to operate the processor at a reduced speed to save energy? Why or why not?
- Possible Answer:
  - $E = R\,t = (k_v V^2 f + R_0)(C/f) = k_v V^2 C + R_0 C/f$
- Conclusion: E is minimum when f is maximum.
  - Operate at top speed
- Is this really true? What are the underlying assumptions?

Reducing Processor Frequency

- Does it make sense to operate the processor at a reduced speed to save energy? Why or why not?
- Possible Answer:
  - $E = R\,t = (k_v V^2 f + R_0)(C/f) = k_v V^2 C + R_0 C/f$
- Conclusion: E is minimum when f is maximum.
  - Operate at top speed
- Is this really true? What are the underlying assumptions?
  - Voltage is constant, independently of frequency
  - Energy is spent only when task is running and is zero otherwise (because we multiplied power by the execution time of the task, C/f , and not more, which implicitly assumes that no energy is spent once task is done)
  - Compute-intensive task (because the execution time was inversely related to CPU frequency, f).

Understanding the Dynamics of Workload

- What if a task is memory intensive?

Understanding the Dynamics of Workload

- What if a task is memory intensive?
  - Then the execution time is a function of memory delay, which is independent of CPU speed. Hence, slowing down the CPU does not have an effect on execution time.
  - $E = R\ t = (k_v V^2 f + R_0)\ T$
- T does not depend on the time

Example - Memory Intensive Task

- The power spent by a processor is given by:
  - $R = 10f + 1$
  - Where f can be 1, 0.5, or 0.3

- If the task is memory intensive and the processor goes to a zero-energy sleep mode once done, what is the energy-optimal frequency?

Example - Memory Intensive Task

- The power spent by a processor is given by:
  - $R = 10f + 1$
  - Where f can be 1, 0.5, or 0.3

- If the task is memory intensive and the processor goes to a zero-energy sleep mode once done, what is the energy-optimal frequency?

- Answer: $f = 0.3$, because energy is proportional to power (since execution time does not depend on processor frequency for memory-intensive tasks), so minimizing f minimizes both power and energy. $E = R\ t$

Example - CPU Intensive Task

- The power spent by a processor is given by:
  - $R = 10f + 1$
  - Where f can be 1, 0.5, or 0.3

- If the task is CPU intensive and the processor goes to a zero-energy sleep mode once done, what is the energy-optimal frequency?

Example - CPU Intensive Task

- The power spent by a processor is given by:
  - $R = 10f + 1$
  - Where f can be 1, 0.5, or 0.3

- If the task is CPU intensive and the processor goes to a zero-energy sleep mode once done, what is the energy-optimal frequency?

- Answer: $f = 1$, because maximizing f in the above energy equation minimizes energy.

DVS - The Critical Frequency

- There exists a minimum frequency below which no energy savings are achieved
  - $R = k_f f^3 + R_0$
  - $E = (k_f f^3 + R_0)(C/f) = k_f f^2 C + R_0 C/f$
  - $dE/df = 2k_f fC - R_0 C/f^2 = 0$

$$f = \sqrt[3]{\frac{R_0}{2k_f}}$$

DVS - Example

- The power consumption of a processor changes with frequency as follows:
  - $R = 8f^3 + 2$
- Which frequency minimizes energy consumption?

DVS - Example

- The power consumption of a processor changes with frequency as follows:
  - $R = 8f^3 + 2$
- Which frequency minimizes energy consumption?
  - Memory intensive use min. Freq.
    - $E = (8\ f^3 + 2)\ (C/f) = 8C\ f^2 + 2\ C/f$

DVS - Example

- The power consumption of a processor changes with frequency as follows:
  - $R = 8f^3 + 2$
- Which frequency minimizes energy consumption?
  - Memory intensive use min. Freq.
    - $E = (8\ f^3 + 2)\ (C/f) = 8C\ f^2 + 2\ C/f$
  - CPU intensive need to find derivative
    - $dE/df = 16Cf - 2C/f^2 = 0,$ -> $f = 0.5$

DVS - Utilization

- At which frequency would you run the aforementioned processor if it is used to schedule CPU-intensive independent periodic tasks using EDF ?

DVS - Utilization

- At which frequency would you run the aforementioned processor if it is used to schedule CPU-intensive independent periodic tasks using EDF and utilization at frequency was 0.7?

- Answer: f=0.7 (so utilization does not exceed 1)

DVS - Utilization

- At which frequency would you run the aforementioned processor if it is used to schedule CPU-intensive independent periodic tasks using EDF and utilization at frequency f was 0.4?

- Answer: f=0.5 (energy optimal as we computed earlier)

DVS - How Many Processors to Use?

- Consider using one processor at frequency f versus two at frequency f/2

- Case 1: Total power for one processor
  - $k_f\, f^3 + R_0$
- Case 2: Total power for two processors
  - $2\ \{k_f\ (f\ /2)^3 + R_0\} = k_f\, f^3 / 4 + 2R_0$
- The general case: n processors
  - $n\ \{kf\ (f\ /n)^3 + R_0\} = k_f\, f^3 / n^2 + n\ R_0$

DVS - How Many Processors to Use? - Example

- The power consumption of a core at frequency f is given by: $64f^3 + 2$. What is the optimal number of cores to use in order to minimize total power consumption? Assume that core frequency can be decreased proportionally to the number of cores when computation is split across multiple cores (while keeping the total execution time fixed).

- Power $= n \{64 (f/n)^3 + 2\} = 64 f^3/n^2 + 2n$
- $dPower/dn = -128 f^3/n^3 + 2 = 0$
- $n = 4 f$

Processor Performance States (P-States)

- P0 max power and frequency
- P1 less than P0, voltage/frequency scaled
- P2 less than P1, voltage/frequency scaled
- ...
- Pn less than P(n-1), voltage/frequency scaled

Processor “Sleep” States (C-states)

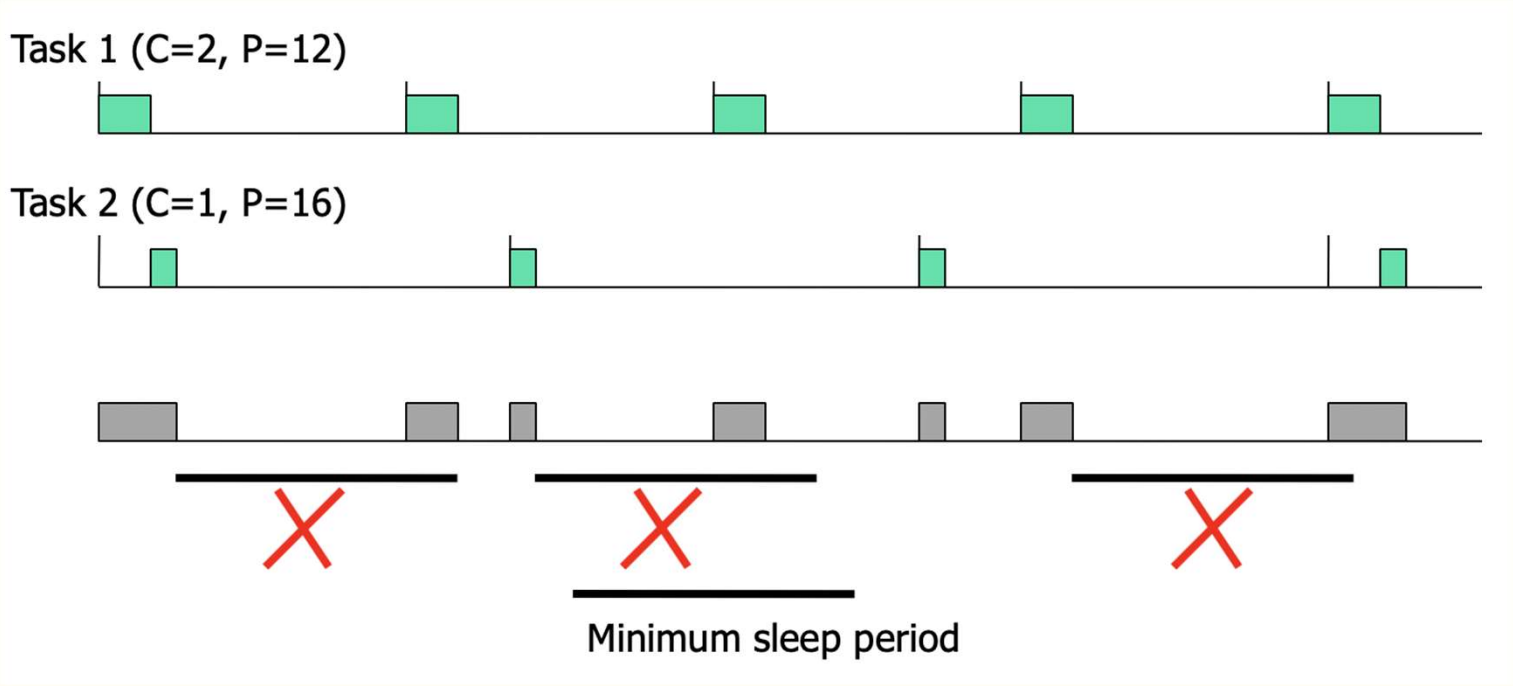
- C0: is the operating state.
- C1 (often known as Halt): is a state where the processor is not executing instructions, but can return to an executing state instantaneously. All ACPI-conformant processors must support this power state.
- C2 (often known as Stop-Clock): is a state where the processor maintains all software-visible state, but may take longer to wake up. This processor state is optional.
- C3 (often known as Sleep) is a state where the processor does not need to keep its cache, but maintains other state. This processor state is optional.

Turning Processors Off - The Cost of Wakeup

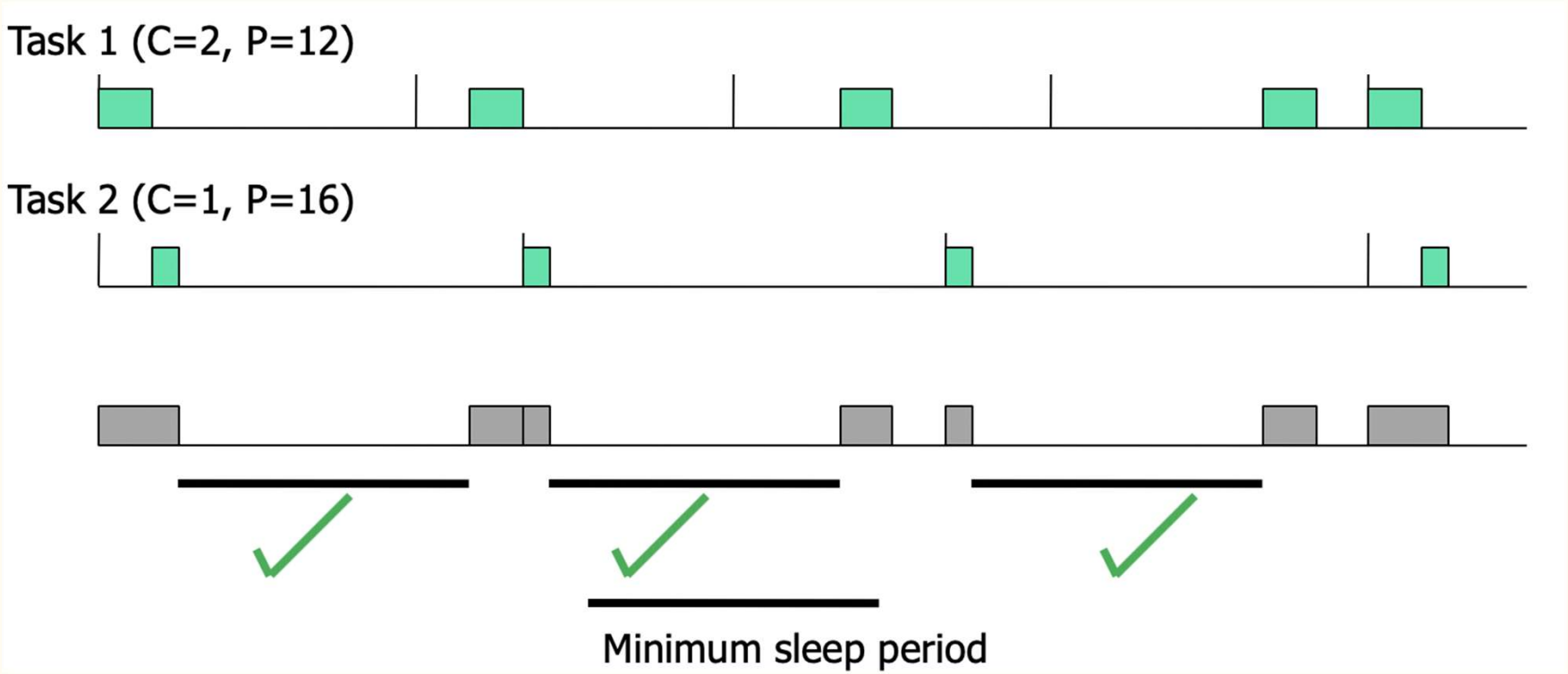
- Energy expended on wakeup, E_{wake}
- To sleep or not to sleep?
  - Not to sleep (for time t): $E_{no\text{-}sleep} = (k_v V^2 f + R_0)\, t$
  - To sleep (for time t) then wake up: $E_{sleep} = P_{sleep}\, t + E_{wake}$
  - To save energy by sleeping: $E_{sleep} < E_{no\text{-}sleep}$

$$t > \frac{E_{wake}}{k_v V^2 f + R_0 - P_{sleep}}$$

DPM and the Problem with Work-conserving Scheduling



DPM and the Problem with Work-conserving Scheduling -Batching to Rescue



To Sleep or Not?

- A processor uses 2.1 Watt when awake and 0.1 Watt when in sleep mode. Wake up cost is 0.5 Joule. What is the minimum sleep period to break even?

To Sleep or Not?

- A processor uses 2.1 Watt when awake and 0.1 Watt when in sleep mode. Wake up cost is 0.5 Joule. What is the minimum sleep period to break even?

- Answer:

  $(2.1 - 0.1)\ t > 0.5$

  Thus, $t > 0.25$ s

Announcements

- Exam will be in class on Thursday during regular class hours
- MP2 demos scheduling sheet will be released today
- MP3 will be posted today