Aperiodic, Multicore, and Distributed Scheduling

Tarek Abdelzaher

The 4th Credit Project (Suggested: 1-2 persons per project)

- Option 1: Develop a 30 min survey presentation on an advanced topic of your choice in real-time and embedded computing.
  - Topic name due 10/17.
  - Slides due 11/17.
  - Presentation the week of 12/2
- Example topics:
  - Self-driving cars: the state of the art and future challenges
  - Real-time AI
  - Multicore scheduling – main challenges and results
  - Embedded system security
  - Scheduling Map/Reduce workflows (with emphasis on time support)
  - Participatory and social sensing (crowd-sensing)
  - Software model checking (proving software correctness)
  - IoT market

The 4th Credit Project (Suggested: 1-2 persons per project)

- Option 2: Implement a real-time or embedded systems service
  - Service name due 10/17.
  - Slides due 11/17.
  - Presentation + Demo the week of 12/2
- Example services:
  - A real-time scheduler for "Intelligence as a Service"
  - Security and diagnostics
  - Disaster response services
  - Social sensing services
  - Your idea here…

Mixed Periodic and Aperiodic Task Systems

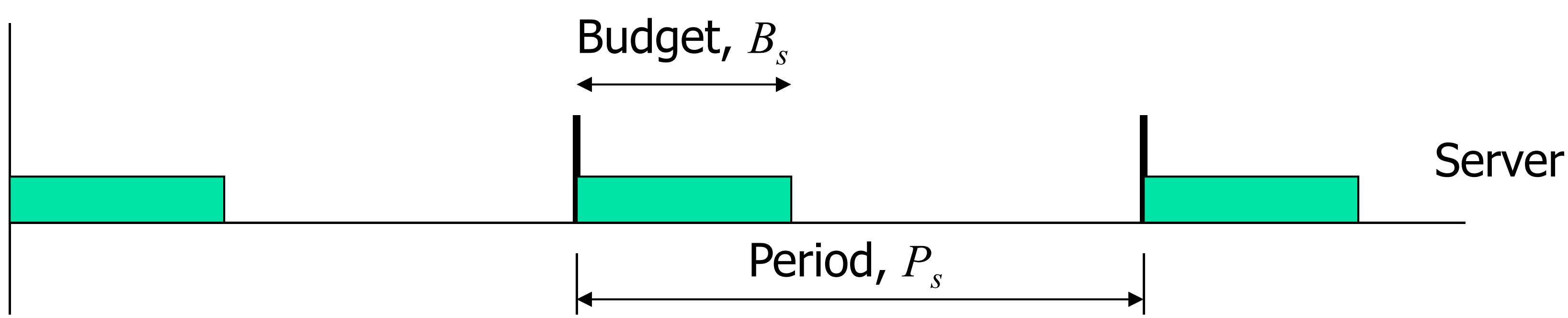
- Question: how to execute aperiodic tasks without violating schedulability guarantees given to periodic tasks?

Mixed Periodic and Aperiodic Task Systems

- Question: how to execute aperiodic tasks without violating schedulability guarantees given to periodic tasks?
- One Answer: Execute aperiodic tasks at lowest priority
  - Problem: Poor performance for aperiodic tasks

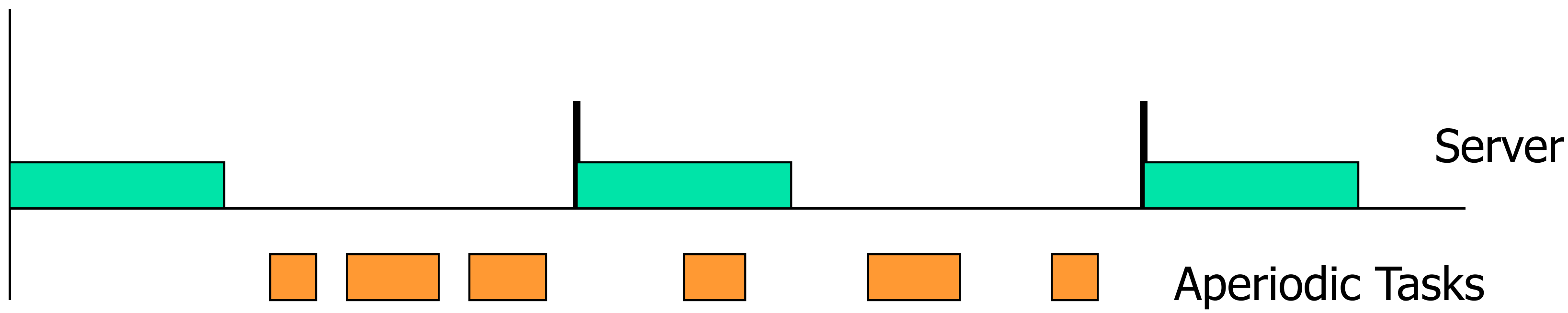
Mixed Periodic and Aperiodic Task Systems

- 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



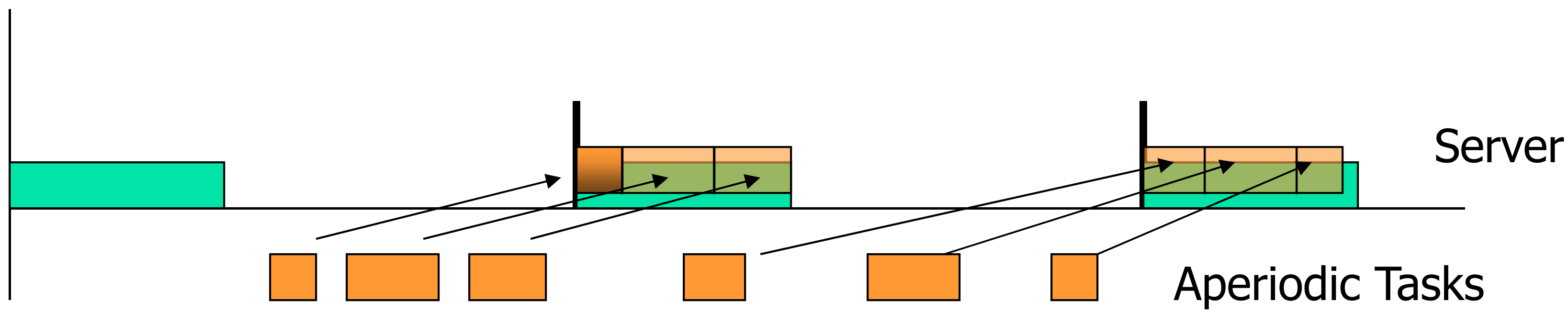
Mixed Periodic and Aperiodic Task Systems

- 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



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

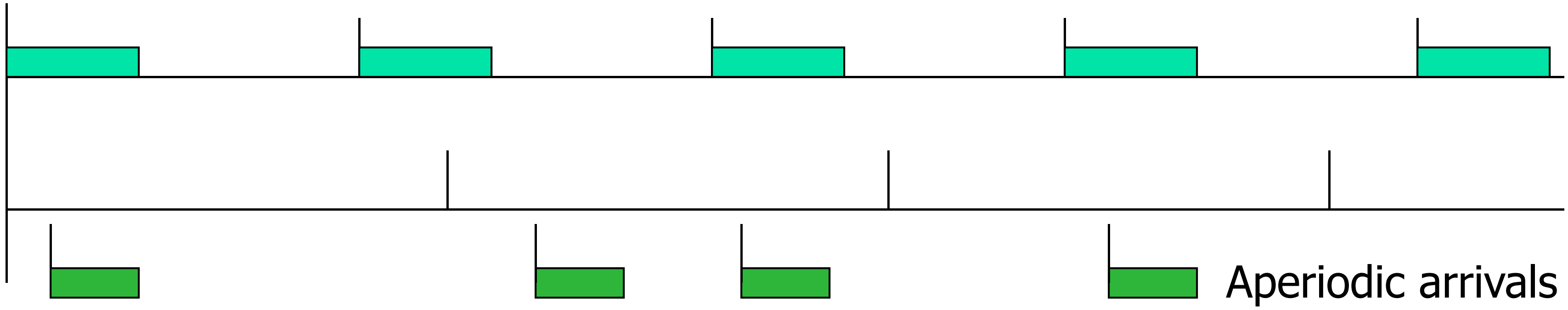


Polling Server

- 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

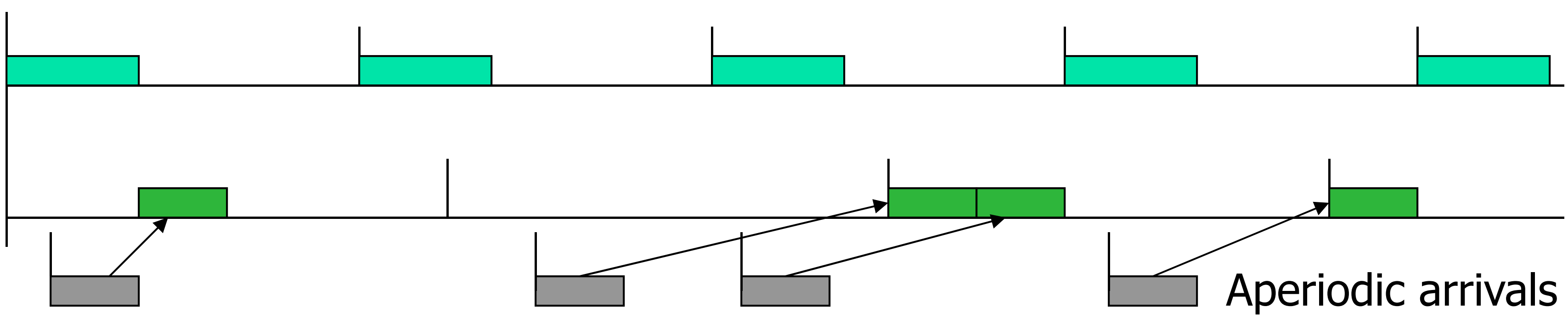
Example of a Polling Server

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



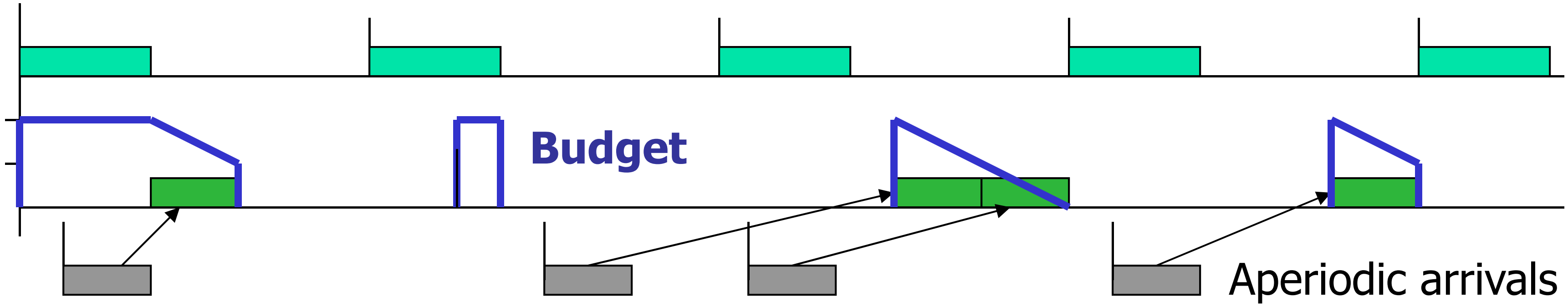
Example of a Polling Server

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



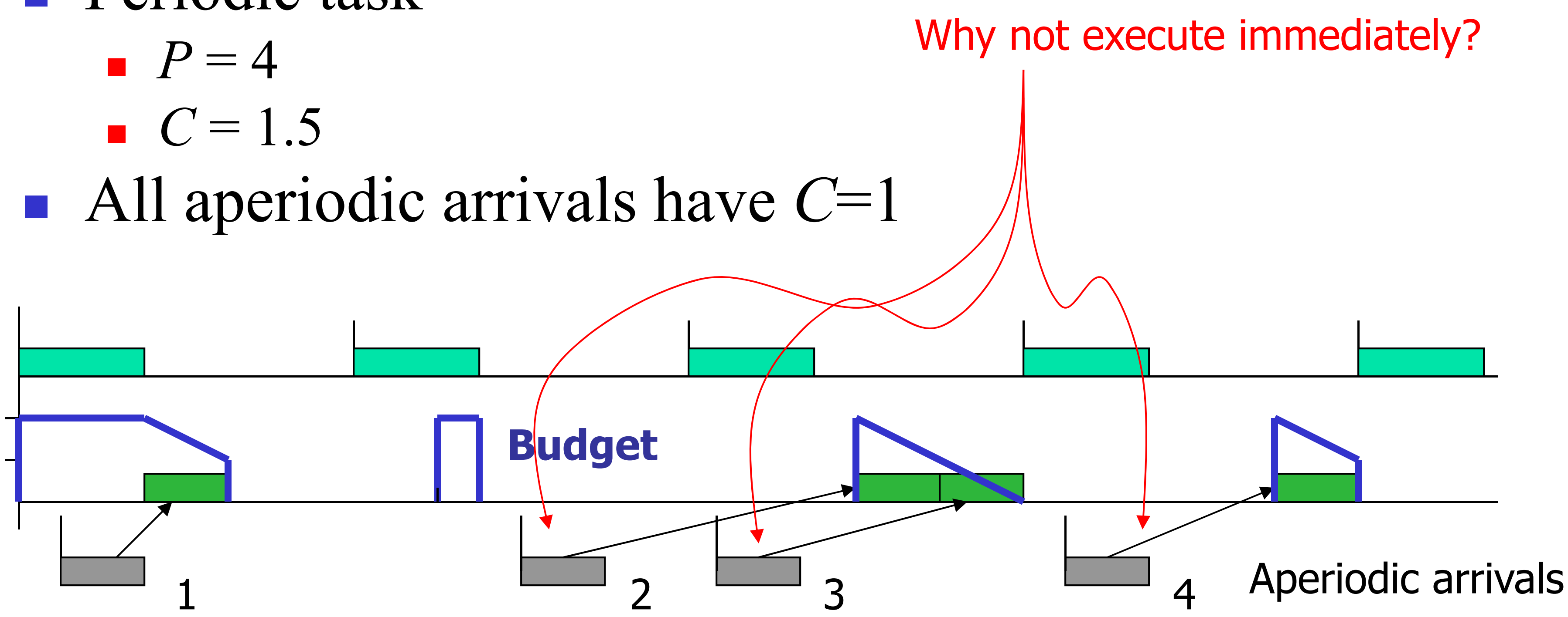
Example of a Polling Server

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



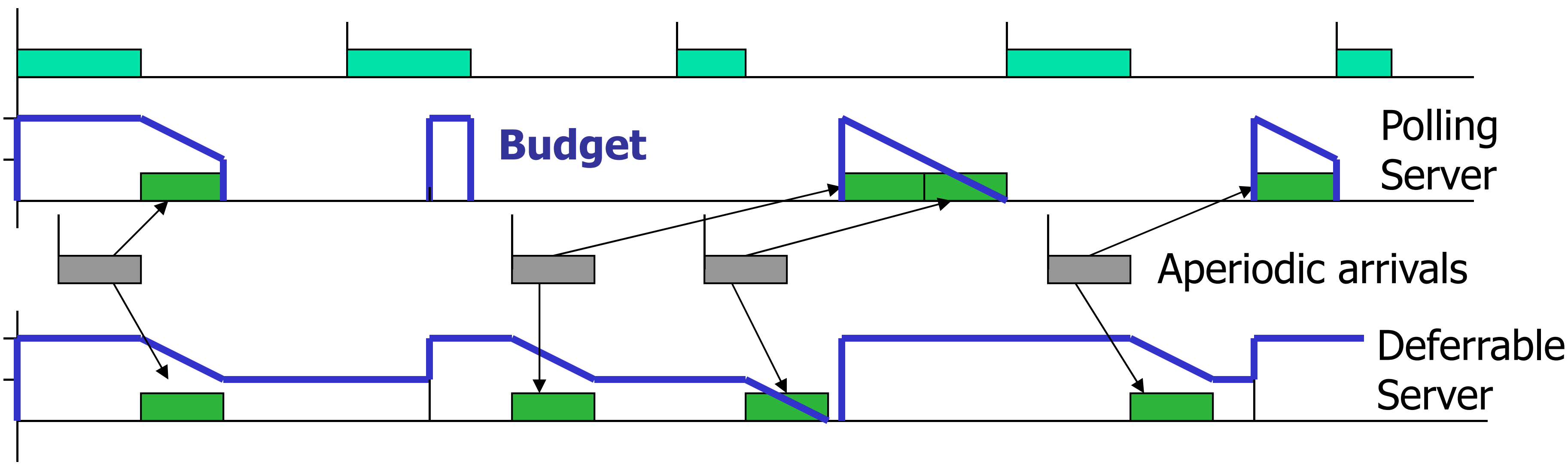
Example of a Polling Server

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

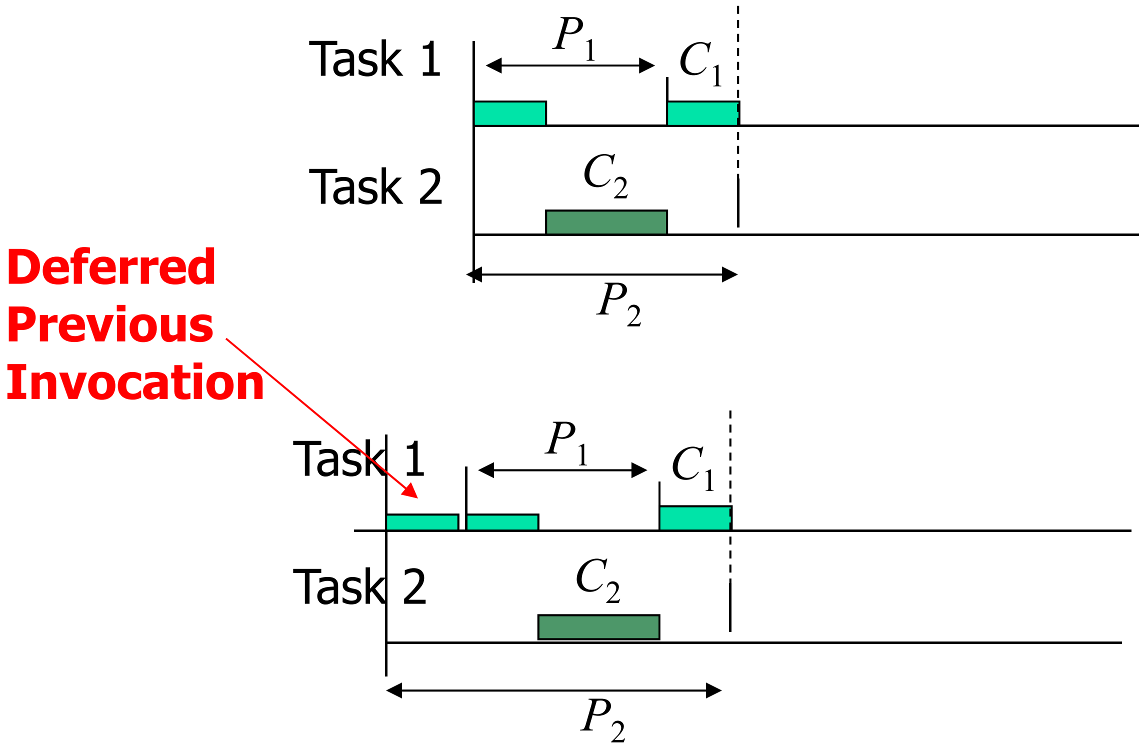


Deferrable Server

- Keeps the balance of the budget until the end of the period
- Example (continued)



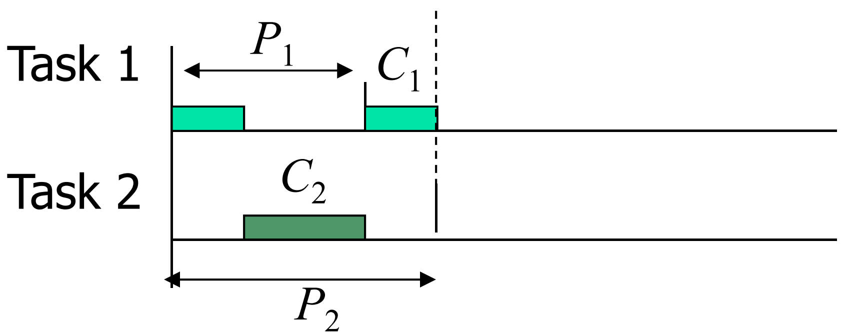
Worst-Case Scenario

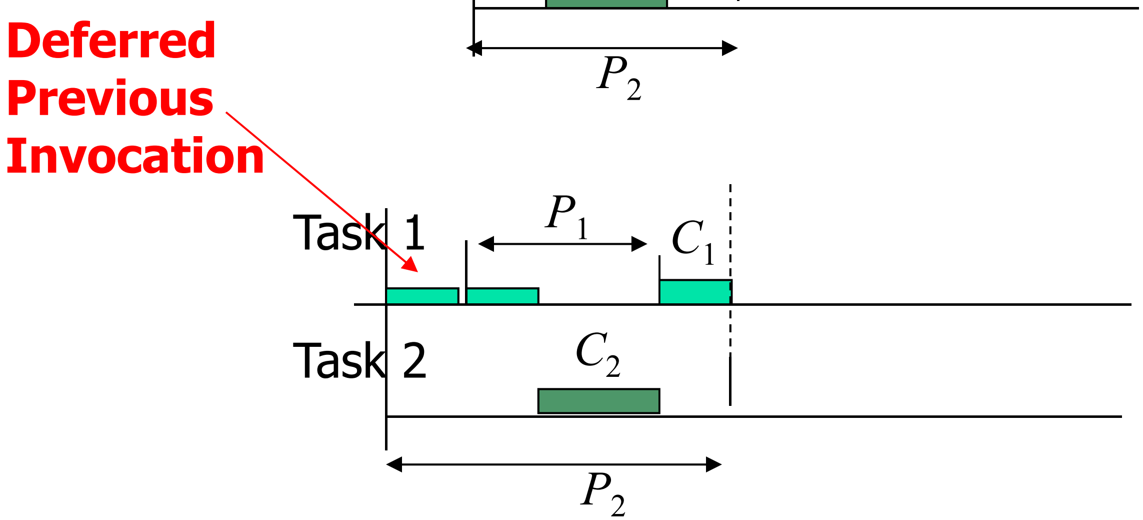


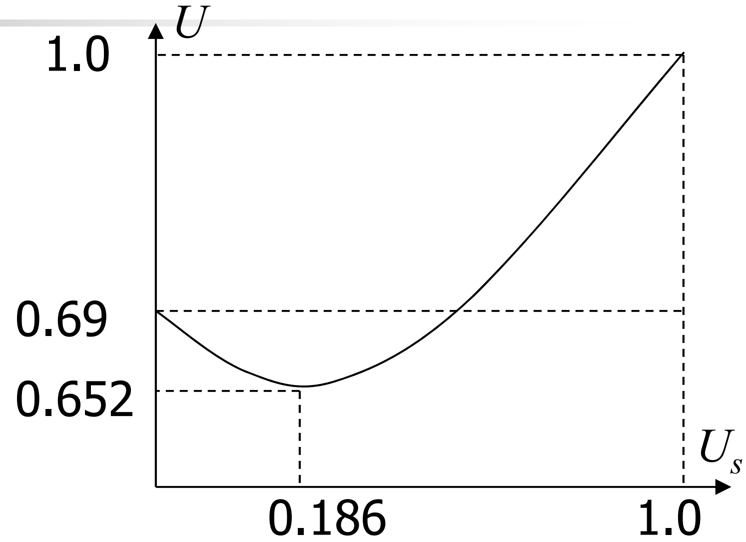
$$U_p \leq \ln\left(\frac{U_s + 2}{2U_s + 1}\right)$$

Exercise: Derive the utilization bound for a deferrable server plus one periodic task

Worst-Case Scenario







$$U_p \le \ln\left(\frac{U_s + 2}{2U_s + 1}\right)$$

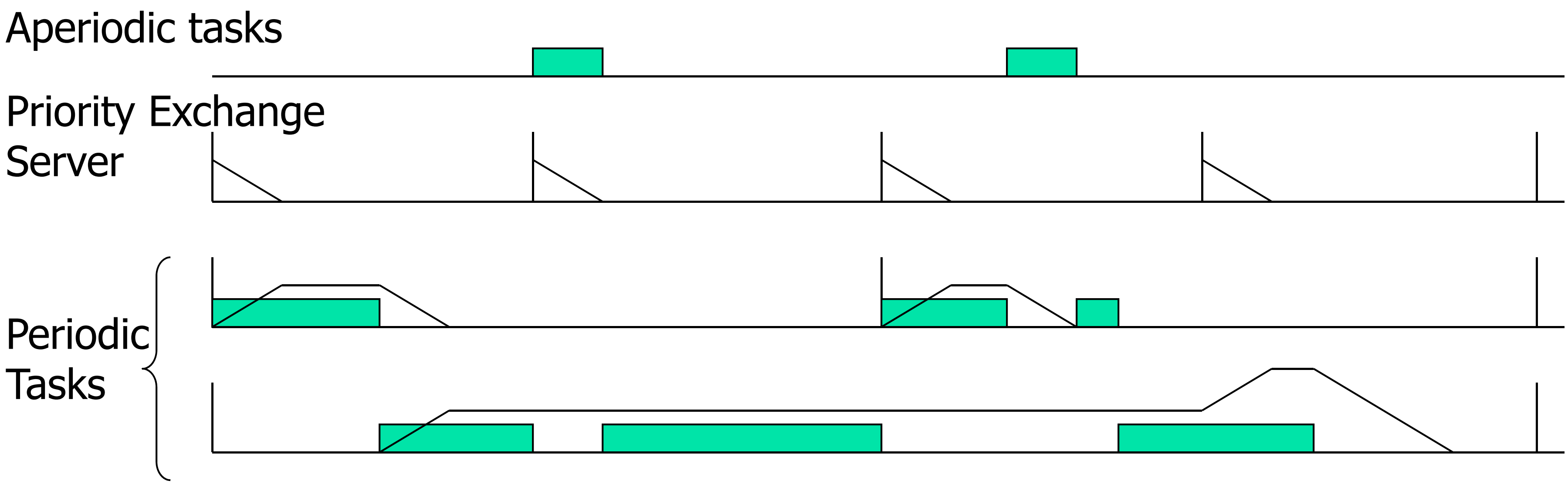
Exercise: Derive the utilization bound for a deferrable server plus one periodic task

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

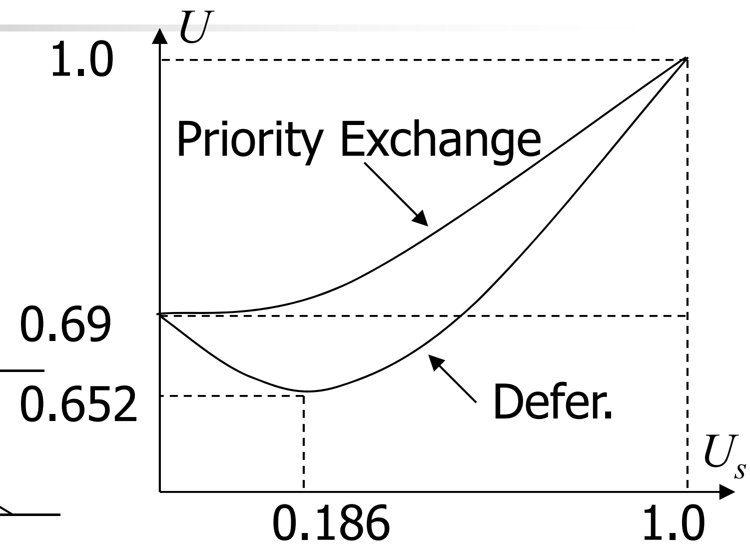
Priority Exchange Server

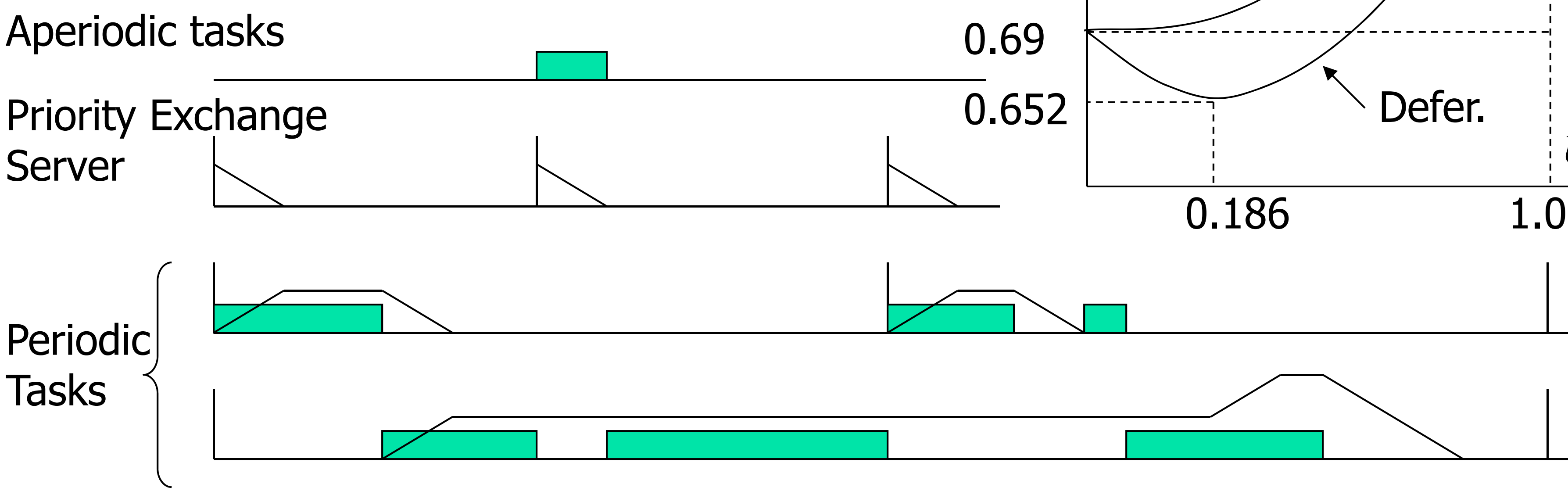
Example



Priority Exchange Server

Example





$$U_p \leq \ln\left(\frac{2}{U_s + 1}\right)$$

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 $[t_A, t_I]$

$$U_p \leq \ln\left(\frac{2}{U_s + 1}\right)$$

Slack Stealing Server

- Compute a slack function $A(t_s, t_f)$ that says how much total slack is available
- Admit aperiodic tasks while slack is not exceeded

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?

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

$$U \leq (m+1)/2$$

(sufficient condition)

Utilization Bound for Partitioned EDF

- There cannot be a better bound than:

$$U \leq (m+1)/2$$

Why?

Utilization Bound for Partitioned EDF

- There cannot be a better bound than:

$$U \leq (m+1)/2$$

Why?

Consider m tasks of utilization (0.5 + very small value) that arrive first, then one more task of utilization = 0.5. Can the last task 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}?

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 set is schedulable if:

$$U \le \frac{\beta m + 1}{\beta + 1},$$

where

$$\beta = \left\lfloor \frac{1}{U_{max}} \right\rfloor$$

Utilization Bound for Partitioned EDF

- $$U \leq \frac{\beta m+1}{\beta+1}, \text{ where } \beta = \left\lfloor \frac{1}{U_{max}} \right\rfloor$$

Why?

Intuition: Imagine all cores are full of tasks of maximum weight (hence, βU_{max} on each core) then a new task arrives that causes the utilization of a core to barely overflow.

Utilization Bound for Partitioned EDF

- $U \le \frac{\beta m+1}{\beta+1}$, where $\beta = \left\lfloor \frac{1}{U_{max}} \right\rfloor$

Why?

Intuition: Imagine all cores are full of tasks of maximum weight (hence, βU_{max} on each core) then a new task arrives that causes the utilization of a core to barely overflow.

$U = 1 + (m-1)\,\beta U_{max} \le 1 + (m-1)\,\beta \frac{1}{\ldots} = \beta m+1 \ldots$

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?

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?

- Task set is schedulable if $U \leq 1$

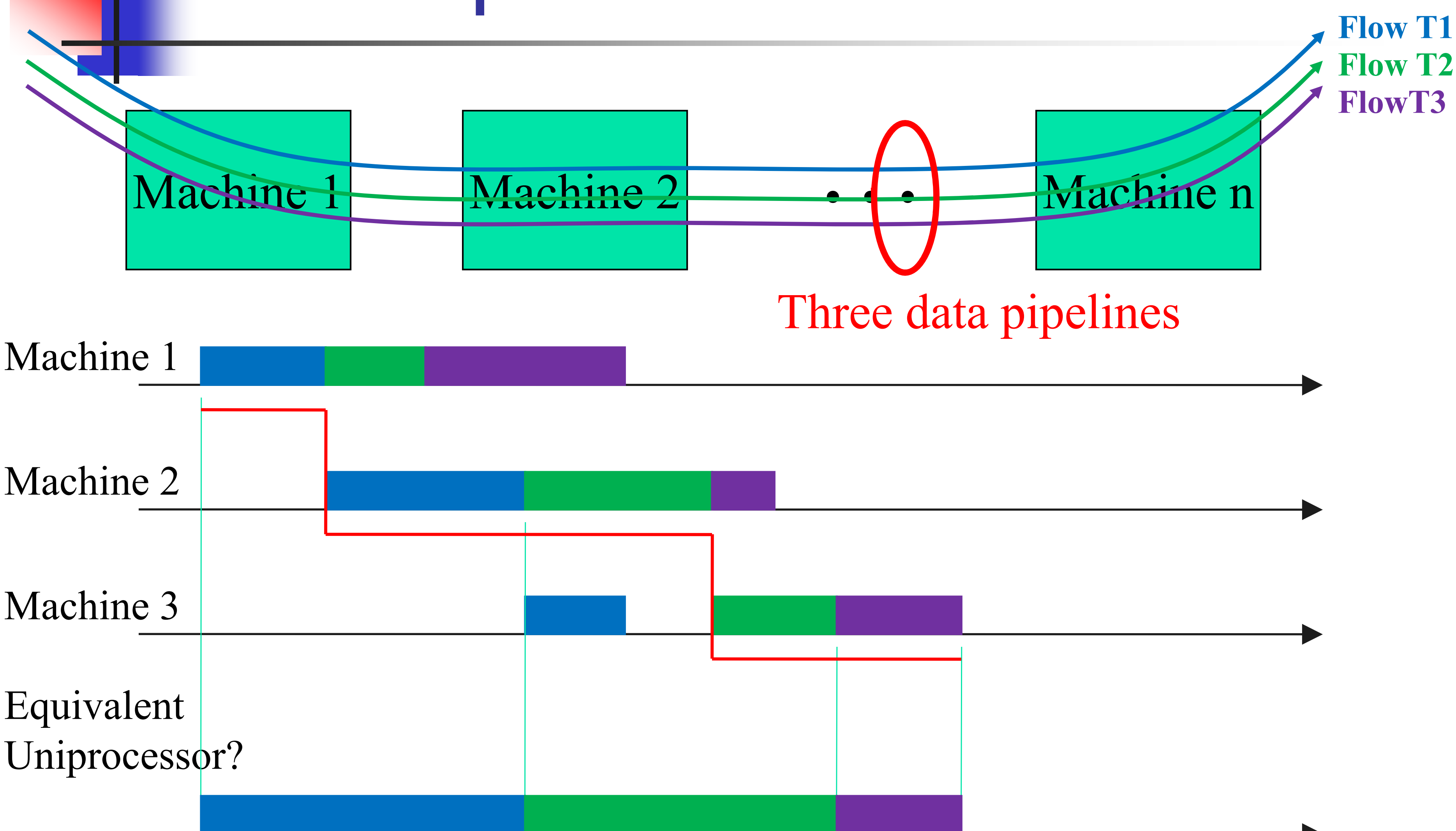
Utilization Bound for Global EDF

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

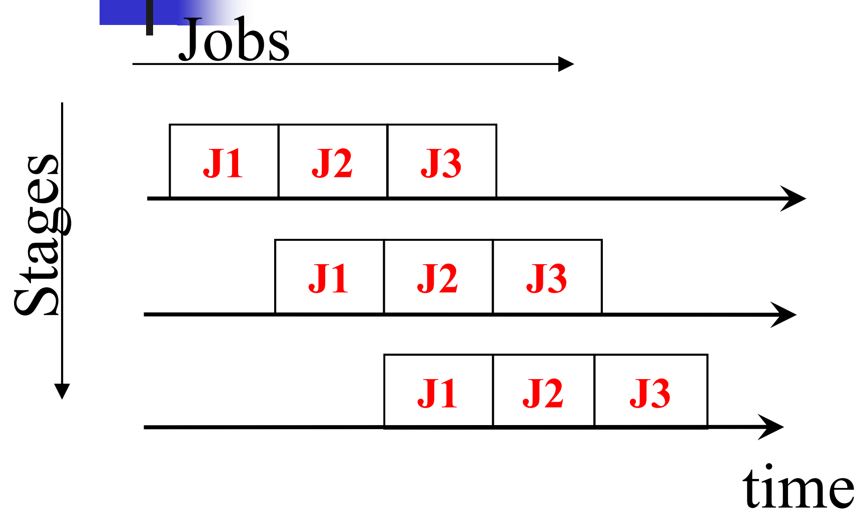
Utilization Bound for Global EDF

- What if maximum task utilization is U_{max}?
- Task set is schedulable if $U \leq m - (m-1)U_{max}$

Task Pipelines



Pipeline (Data) Processing

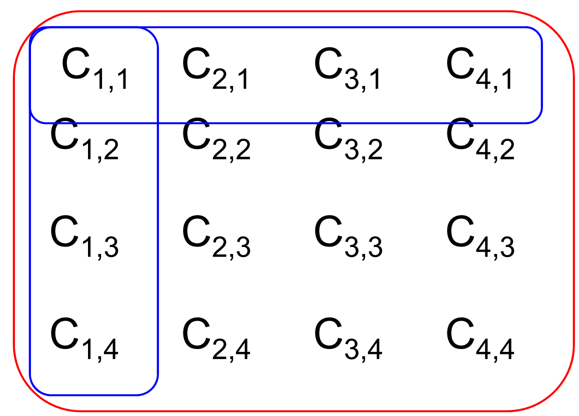


- Sub-additive delay composition due to pipeline overlap

$$\textbf{\textit{Delay}} < \sum_{\textit{stages}} \sum_{\textit{tasks}}$$

- Especially useful for systems with tight deadlines

$$\text{Delay} \leq \sum_{\text{allstages}} \max_{\substack{\text{higher}\\ \text{priority}\\ \text{jobs}}} C_{i,j} + \sum_{\substack{\text{higher}\\ \text{priority}\\ \text{jobs}}} 2 \max_{\text{allstages}} C_{i,j}$$



Main Results

- Consider task n that traverses stages j = 1, …, N together with higher priority tasks i.
- Delay composition theorem:
  - Let $MaxNode_j$ be the longest execution time of all tasks that execute on node j.
  - Let $MaxTask_i$ be the maximum execution time of task i across all nodes visited by the task
- The delay of task n is given by:

$$\text{Delay} < \sum_{i>n} (2\ \text{MaxTask}_i) + \sum_j \text{MaxNode}_j$$

Task Set Reduction

- Each higher priority task i is reduced to a uniprocessor task with a computation time = 2 MaxTask_i
- Task n (under consideration) is reduced to a unirpcessor task with a computation time = $\sum_j \text{MaxNode}_j$
- Uniprocessor bounds then apply.