Constant Utilization and Total Bandwidth Servers

郭大維 教授

ktw@csie.ntu.edu.tw

嵌入式系統暨無線網路實驗室

(Embedded Systems and Wireless Networking Laboratory)

國立臺灣大學資訊工程學系

All Rights Reserved; Tei-Wei Kuo, National Taiwan University
Source: Jane W.S. Liu, "Real-Time Systems," Prentice Hall, 2000.

Schedulability of Sporadic Jobs in Deadline-Driven Systems

- Definition Revisiting:
  - Tasks versus Jobs
  - Earliest Deadline First (EDF) Scheduling
- Definition: The *density* of a sporadic job with an arrival time r_i, maximum execution time e_i, and the absolute deadline d_i is $e_i/(d_i-r_i)$.
- Definition: A job is *active* in its interval $(r_i,d_i]$.
- Theorem 7.4 [Jane Liu, RTS2000] A system of independent preemptable sporadic jobs is schedulable by EDF if the total density of all active jobs is no greater than 1 at all times.

  (A sufficient condition)

Schedulability of Sporadic Jobs in Deadline-Driven Systems

- Definition: The *instantaneous utilization* of a sporadic job $S_{i,j}$ is $u_{i,j} = e_{i,j} / p_{i,j}$, where $p_{i,j}$ denotes the release-time difference between $S_{i,j}$ and $S_{i,j+1}$.
- Definition: The *instantaneous utilization* of a sporadic task S_i is $u_i = \max_j(e_{i,j} / p_{i,j})$.
- Corollary 7.5 [Jane Liu, RTS2000] A system of n independent preemptable sporadic tasks is schedulable by EDF if the total *instantaneous utilization* of all active jobs is no greater than 1, where the relative deadline of every job is equal to its period (i.e., the release-time difference).

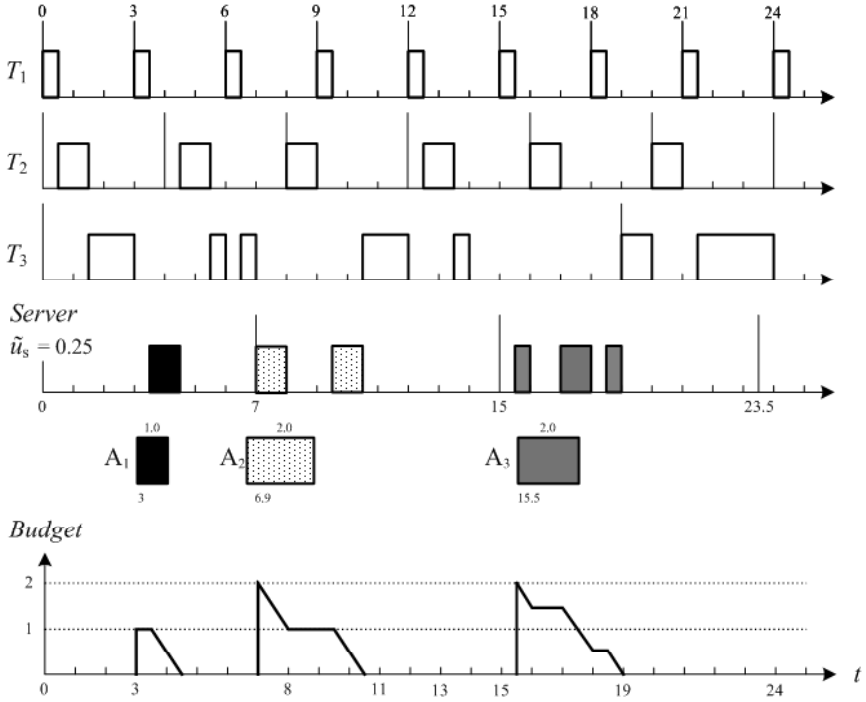
  (An extension to the considerations of periodic tasks)

Constant Utilization Server

- A constant utilization server is defined by its instantaneous utilization u_s, where the scheduling of periodic tasks and servers is done by EDF.
  - Consumption Rule: A server consumes its budget only when it executes for some sporadic job.
  - Replenishment Rule: (e_s: budget; d: deadline)
    1) Initially, $e_s = 0$ and $d=0$.
    2) When a sporadic job with execution time e arrives at time t to an empty queue,
       a) If $t < d$, do nothing; otherwise, $d = t + e/u_s$, and $e_s=e$.
    3) At the deadline d of the server,
       a) If the server is backlogged, set the server deadline to $d + e/u_s$, and $e_s=e$.
       b) If the server is idle, do nothing
- A constant utilization server is always given enough budget to complete the job at its queue head each time its budget is replenished.

Constant Utilization Server

- A1: d=3+1.0/0.25=7; A2: At 7, d=7+2/0.25=15; A3: d=23.5



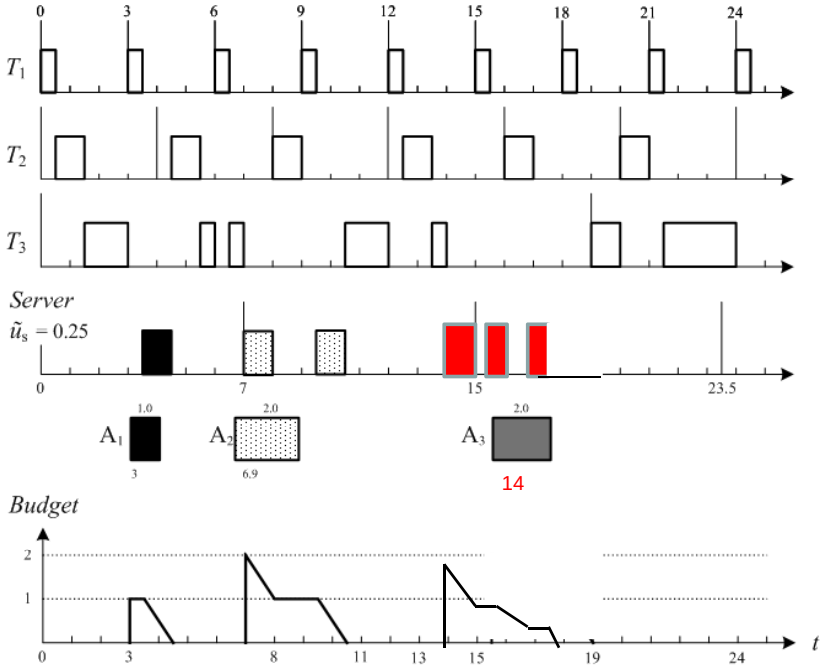
* There is no deadline missing for all periodic tasks if the total size of constant bandwidth servers is no larger than (1-Δ) if Δ is the total density (ei/min(di,pi)) of periodic tasks.

Total Bandwidth Server

- Motivation: Improve a constant utilization server by allowing a server to claim the unused background time.
  - Consumption Rule: A server consumes its budget only when it executes for some sporadic job.
  - Replenishment Rule: (e_s: budget; d: deadline)
    1) Initially, e_s =0 and d=0.
    2) When a sporadic job with execution time e arrives at time t to an empty queue, set d to max(d,t) + e/u_s, and e_s=e.
    3) When the server completes the current aperiodic job, the job is removed from the queue.
       a) If the server is backlogged, set the server deadline to d + e/u_s, and e_s=e.
       b) If the server is idle, do nothing

Total Bandwidth Server

- A1: d=3+1.0/0.25=7; A2: At 7, d=7+2/0.25=15; A3 now arrives at 14. At 14, the d is 15, and the new d = 23.5. A3 completes at 17.5, instead of 19 because A3 now uses the processor was idle in (14,15).



Total Bandwidth Server

- Corollary 7.7 [Jane Liu, RTS2000] When a system of independent, preemptable periodic tasks is schedulable with one or more total bandwidth and constant utilization servers on the EDF basis, every periodic task and every server meets its deadline if the sum of the total density of periodic tasks and the total size of all servers is no greater than 1.
- Corollary 7.8 [Jane Liu, RTS2000] When a system of independent, preemptable periodic tasks is schedulable with one or more total bandwidth and constant utilization servers on the EDF basis, every periodic task and every server meets its deadline if the sum of the total density of periodic tasks and the total size of all servers is no greater than $1 - b_{max}(np)/D_{min}$, where $b_{max}(np)$ and D_{min} denote the maximum nonpreemptable duration and the min(relative deadlines, sporadic job execution time), respectively.

Fairness and Starvation

- Fairness: The fraction time of the processor time in the interval attained by each server that is backlogged throughout the interval is proportional to the server size.
  - Total bandwidth servers could be unfair:
    - The size of TB1/TB2 is 0.5;
    - TB1 remains backlogged in (0, t), but TB2 is idle.
    - By t, TB1 have executed for t time units, and its deadline is at least 2t.
    - Between t and 2t, jobs with execution time less than t arrives and keep TB backlogged; The deadline of TB2 is less than TB1
    - As a result, it is unfair in (t,2t) even though it is fair in (0,t), where t can be any number.

Fairness and Starvation

- Fairness Revisiting: The fraction time of the processor time in the interval attained by each server that is backlogged throughout the interval is proportional to the server size.
  - A scheduler is fair in the interval (t_1, t_2) if the normalized service $w_i(t_1,t_2)/u_i$ attained by all servers that are backlogged during the interval differ no more than the fairness threshold FR>=0, where $w_i(t_1,t_2)$ is the attained service time of the i-th server.

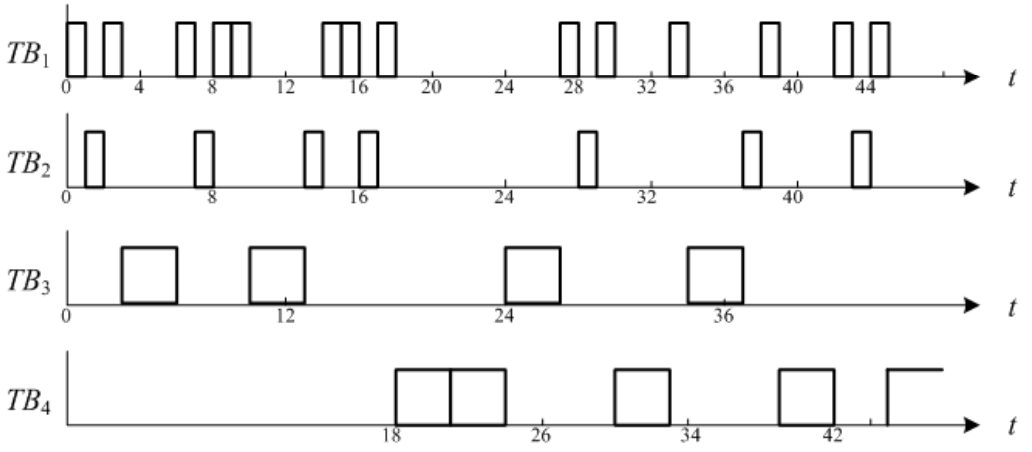
Ideally, $\frac{w_i(t_1,t_2)}{w_j(t_1,t_2)} = \frac{u_i}{u_j}$ Or, $w_i(t_1,t_2) = u_i (t_2-t_1)$

Fairness and Starvation

- Why Starvation?
  - It is because the background time is made available to some total bandwidth server, e.g., TB1 - It is also the difference between total bandwidth servers and constant bandwidth servers.
    - An observation: The current deadline of a constant bandwidth server of size u_i is never more than $(\max e_i)/u_i$. In other words, the length of starvation suffered by any server is bounded by $\max_i((\max e_i)/u_i)$.

Fairness and Starvation

- Example: $u_1=1/4$; $u_2=1/8$; $u_3=1/4$; $u_4=3/8$;
  - First arrival: A1(t=0,e=1); A2(0,1); A3(0,3); A4(18, 3)
  - Have servers kept backlogged continuously.
  - At 18, the deadlines of TB1, TB2, and TB3 are 36, 40, 36, respectively, and they share the 18-unit service time non-proportional to their sizes (i.e., 8, 4, and 6).



Fairness and Starvation

- Example: u_1=1/4; u_2=1/8; u_3=1/4; u_4=3/8;
  - First arrival: A1(t=0,e=1); A2(0,1); A3(0,3); A4(18, 3)
  - Have servers kept backlogged continuously.
  - After 18, the normalized services of all servers are identical in time intervals of length 24 or more. Before 18, CU1, CU2, and CU3 executes for 6, 3, and 9 units, respectively. → No starvation and no fairness guarantee