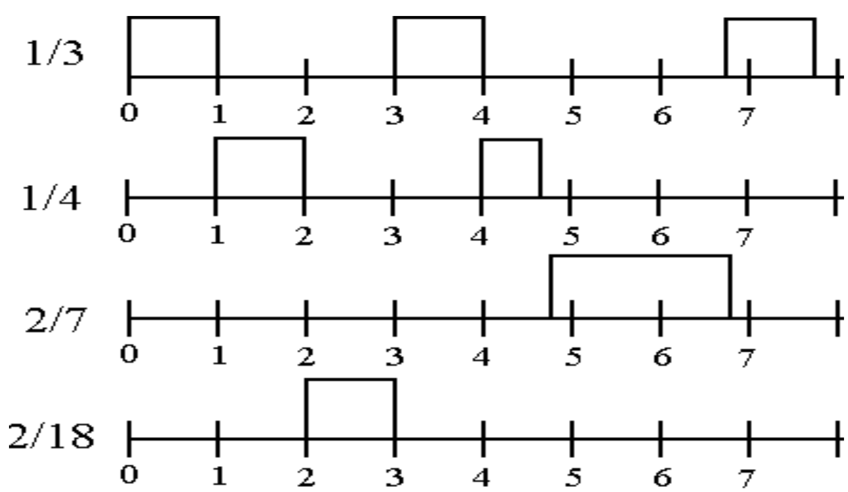
EDF + Aperiodic Jobs

- Under the EDF system, we want to be able to execute aperiodic jobs
  - Suppose we have three periodic tasks, 1.1/3, 1/4, 2/18, and an aperiodic server of 2/7
    - If an AP job arrives at 4.8, can we make its deadline?
    - Can we schedule it using RM?



Weighted Round Robin

- EDF has some similarities to Round Robin
- In regular Round Robin:

1	1	1	1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	1	2

- In a Weighted Round Robin:

1	1	2	2	2	3	4	4	5	6

  - Everyone has a different weight, but we still rotate through everyone
  - Your "share" of the system is $u_i = \frac{w_i}{\sum w_i}$

GPS – Generalized Processor Sharing

- Everyone has the same period, and individual jobs have different timeslots
- $$S = \frac{u_i}{\sum u_i}$$
  - Let $J_1=3, J_2=5, J_3=6, J_4=2$

2/ 16	3 / 16	5 / 16	6 / 16

WFQ – Weighted Fair Queuing

- Each channel of a 10 Mbps cable is given a different rate
- The finish time for each channel will be different.
  - 6/16 channel will finish first

$$10 \text{ MBps} \begin{cases} 30/16 \quad \text{MBps} \qquad 10\text{k packet} \\ 50/16 \quad \text{MBps} \qquad 10\text{k packet} \\ 60/16 \quad \text{MBps} \qquad 10\text{k packet} \\ 20/16 \quad \text{MBps} \qquad 10\text{k packet} \end{cases}$$

WFQ

- Packets are still sent in an EDF fashion.
- Finish times for each channel are 1/rate:
  - 5.3, 10., 15.9   1
  - 3.2, 6.4, 9.6, 12.8   2
  - 2.6, 5.2, 7.8, 10.4   3
  - 8, 16, 24, 32   4

3	2	3	1	2	3	4	2	3	1

WFQ

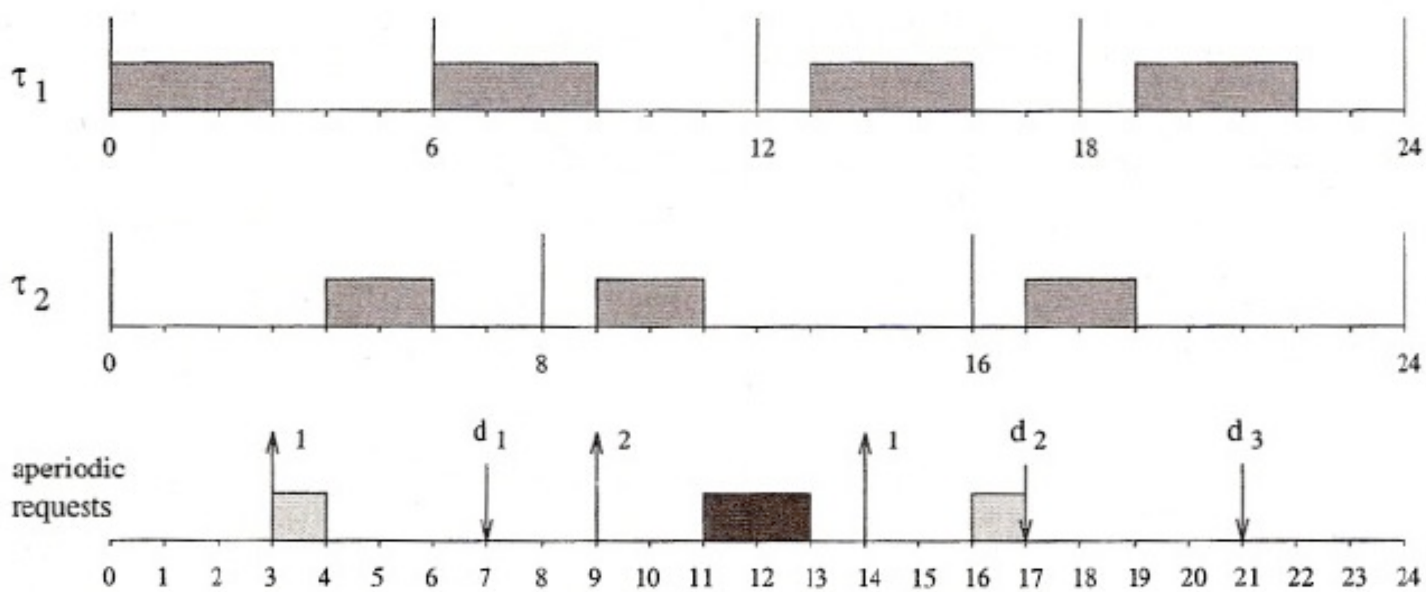
- The sequence yields a density queue
- Now, instead of keeping track of the period, we define the share and deadline.
- A similar idea applies to the Total Bandwidth Server

Total Bandwidth Server

- Utilization U_s,
  - Initially, $e_s = 0$, $d = 0$
  - If an aperiodic job arrives with e at t
    - *Assign job with* $d = \max(t, d) + e/u_s$ *and* $c_s = e$
  - If aperiodic jobs have different execution times, their deadlines will be different
    - An aperiodic jobs of (1,7) that arrives at 4.8 has until 11.8 to complete
    - An aperiodic jobs of (2,7) that arrives at 4.8 has until 18.8 to complete

Total Bandwidth Server Example

- Two periodic tasks (3,6), (2,8).
  - *Up* = 0.5 + 0.25 = 0.75
- TBS with utilization *Us* = 1 - *Up* = 0.25.
- The first aperiodic (*e1*=1) arrives at *t* = 3
  - deadline *d1 = r1 + e1/Us* = 3 + 1/0.25 = 7.
- The second (*e2*=2) arrives at time *t* = 9
  - deadline *d2 = r2 + e2/Us* = 17



Issues with TBS

- The server will grant all budget as requested
  - If an aperiodic job asks for more time, its deadline will be farther off and the job will be less responsive
  - Or if a job uses more time than it thinks it needs, other jobs will be delayed and may miss deadline
  - So, a job needs to know how much time it needs
    - If a job doesn't ask for enough time, it won't complete
    - Too much, and the job isn't as responsive

Constant Bandwidth Server

- Constant Bandwidth Server (CBS) tries to correct this problem.
- Idea: you shouldn’t be penalized for wanting more execution time
  - In RM and EDF, priority is based on period.
  - Want a fixed priority and EDF, with a fixed “share”

Constant Bandwidth Server

- We still want to preserve the utilization percentage of periodic and sporadic tasks
  - If (Q_s, P_s) is the budget and period of the sporadic job for the server, respectively, then Q_s / P_s gives the utilization
  - Every time you get the same budget, Q_s
  - If $e > Q_s$, go back to waiting until you have more budget

Constant Bandwidth Server

- server S is defined by $u_s = Q_s / P_s$
- Initially, d_s=0, c_s=0
- J_i arrives at t to S
  - If server S is busy, J_i is placed on S's queue (FIFO)
  - If server S is free
    - if $c_s \geq (d_s - t)u_s$, then $d_s = t + P_s$, $c_s = Q_s$
    - else use d_s, c_s as is

Constant Bandwidth Server

- c_s is decreased by the amount used by any aperiodic job
- Wherever c_s=0 and there is a job waiting/unfinished,
  - Set $c_s = Q_s$, $d_s = d_s + P_s$
  - Budget is replenished and deadline is updated

CBS Example

When a job $J_{i,j}$ arrives and the server is idle, if $c_s \geq (d_{s,k} - r_{i,j})U_s$ the server generates a new deadline $d_{s,k+1} = r_{i,j} + T_s$ and c_s is recharged at the maximum value Q_s, otherwise the job is served with the last server deadline $d_{s,k}$ using the current budget.