Real Time Operating Systems

The Sporadic Server

Luca Abeni

abeni@dit.unitn.it
Sporadic and Aperiodic Tasks

- Arrival times are not known
- Execution times might be unknown too...
- Performing an admission test might be difficult...

How to schedule aperiodic and sporadic tasks?
- Risk to cause deadline misses in other tasks
- Sporadic tasks can be scheduled as periodic tasks with period equal to the Minimum Interarrival Time ⇒ system underutilisation
Serving Tasks with Unknown Utilisation

- Idea: “reshape” aperiodic and sporadic tasks to force their utilisation so that they do disturb other tasks

- Traditional solution: a periodic real-time task (called server) serves aperiodic and sporadic requests
  - Emulated by some scheduling algorithms
  - For this reason, the name of such algorithms often contains the word “Server”

- Server algorithms for aperiodic tasks are often based on static priorities
  - They can coexist with RM or DM
  - Examples: Polling Server, Deferrable Server, Sporadic Server, ...

- Can be modified to work with EDF
The Sporadic Server

- Class of algorithms to serve aperiodic/sporadic tasks
  - Many different implementations have been proposed
  - A sporadic server *tries* (if possible) to serve requests as soon as they arrive (this is why it is called **sporadic**)
  - Has the worst-case behaviour of a periodic task

- A server is described by two parameters $Q^s$ and $T^s$
  - Like other algorithms, is based on a **budget** $q^s$
  - The budget is decreased when the served task executes
  - The budget is *recharged* after $T^s$ from its usage

- The various implementations differ in these **accounting** and **replenishment** rules
Sporadic Servers: Possible Choices

- When / How to replenish?
  - Replenishment can be performed by *always replenishing* $q^s$ to $Q^s$. In this case, $q^s$ is decreased when the highest priority tasks are idle and $q^s > 0$.
  - If $q^s$ is decreased only when the served task is executing, then the budget must be recharged “in chunks”: if $c$ time units are consumed from $t_0$ to $t_0 + c$, then the budget is recharged at time $t_0 + T^s$ as $q^s = q^s + c$

- Can background time be used?
- Can the server use the time unused by higher priority tasks?
Replenishing to $q^S$

$Q^s = 3$

$T^s = 5$

Missed Deadline!
Replenishing in Chunks

budget

0.5 5 0.5
Sporadic Server in POSIX

- POSIX defines various scheduling policies:
  - SCHED_FIFO
  - SCHED_RR
  - SCHED_SPORADIC
  - SCHED_OTHER

- **SCHED_SPORADIC** is a *Sporadic Server*
  - Specific Sporadic Server definition by POSIX
  - The `sched_param` structure must be extended...
  - Difference respect to “traditional” sporadic servers: when the budget is exhausted, the task is not blocked (but is scheduled at a lower priority)
  - Performs replenishments “in chunks”
Sporadic Server Interface

- `struct sched_param` has been extended:
  - `sched_ss_init_budget`: maximum budget $Q^s$
  - `sched_ss_repl_period`: replenishment period $T^s$
  - `sched_ss_low_priority`: background priority (at which the server is scheduled when the budget $q^s$ is depleted)
  - `sched_ss_max_repl`: maximum number of pending replenishments

- The priority of the server is given by `sched_priority`
- When $q^s > 0$, served tasks are scheduled at priority `sched_priority`
POSIX Algorithm

- The budget $q^s$ is decreased when served tasks execute.
- Activation time: job arrival time $r_{i,j}$, or replenishment time when $q^s$ becomes $> 0$.
- Replenishment times are set to $activation\_time + T^s$.
- Replenishment amounts are computed when a job finishes (task blocks), or when the budget is depleted ($q^s = 0$).
- When $q^s = 0$, the task is scheduled at $sched\_ss\_low\_priority$.
- The budget $q^s$ is always $\leq Q^s$.
- Limit the maximum number of pending replenishments: if $sched\_ss\_max\_repl$ replenishments are pending, schedule at $sched\_ss\_low\_priority$. 