Embedded Systems Summary

Nicolas Trüssel

July 31, 2016

1 Miscellaneous

Dependable means

- Reliable (continue working correctly when it worked correctly before)
- Maintainable (recover from errors)
- Available (probability that it's working)
- Safety
- Security

Efficient in terms of:

- Energy
- Code size
- Memory consumption
- Run time
- Weight
- Cost

2 Generic Time Triggered Cyclic Executive Scheduler

Let f denote the frame length, P the full period, $D(k)$ the relative deadline of task k, $C(k)$ its execution time, and $p(k)$ the period of task k (how often it occurs). Then the following conditions have to be satisfied:

- $\forall k. f \leq p(k)$ (at most one execution within a frame)
- $P = \operatorname{lcm}_k (p(k))$
- $\forall k. f \geq C(k)$ (processes start and complete within single frame)
- $\forall k. 2f - \gcd(p(k), f) \leq D(k)$ (between release time and deadline of every task there is at least one frame boundary)

3 Aperiodic Scheduling

	Equal arrival, non-preemptive	Arbitrary arrival, preemptive
Independent Tasks	EDD	EDF
Dependent Tasks	LDF	EDF*

3.1 Definitions

- $J = \{j_1, j_2, \dots, j_n\}$ is a set of tasks.
- a_i or r_i is the arrival / release time of task i.
- d_i is the deadline of task i.
- C_i is the total computation time of task i.
- $c_i(t)$ is the the remaining execution time of task i at time t.
- s_i is the start time of task i.
- f_i is the finish time of task i.
- $L_i = f_i - d_i$ is the lateness of task i.
- $E_i = \max(0, L_i)$ is the exceeding time or tardyness of task i.
- $X_i = d_i - a_i - C_i$ is the laxity or slack of task i.

3.2 EDD

Schedule the tasks in order of non-decreasing deadlines. This minimizes the maximal lateness.

3.3 EDF

Always execute the task with the earliest absolute deadline. Schedulability test:

$$\forall i \in [n].t + \sum_{k=1}^{i} c_k(t) \le d_i$$

3.4 LDF

Among all tasks without successors select the task with the latest deadline. Put it in a stack. Repeat until no more tasks. Now execute tasks as they are on the stack.

3.5 EDF*

Modify arrival and deadline of each task and use EDF on modified tasks.

$$r_j^* = \max_j \left(r_j, \max_i \left(r_i^* + C_i | J_i \to J_j \right) \right)$$

$$d_i^* = \min_i \left(d_i, \min_j \left(d_j^* - C_j | J_i \to J_j \right) \right)$$

4 Periodic Scheduling

Periodic scheduling is always preemptive.

	Deadline = Period	Deadline $\leq$ Period
Static Priority	Rate Monotonic	Deadline Monotonic
Dynamic Priority	EDF	EDF*

4.1 Definitions

- $\Gamma = \{\tau_1, \tau_2, \dots \tau_n\}$ is set of periodic tasks.
- $\tau_{i,j}$ is the j-th instance of task τ_i.
- $a_{i,j}, r_{i,j}, d_{i,j}, s_{i,j}, f_{i,j}$ are the same as for aperiodic tasks.
- Φ_i is the phase of task i (release time of the first instance).
- D_i is the relative deadline of task i.
- T_i is the period of the task (time between 2 releases).

The following hypotheses are assumed

- $r_{i,j} = \Phi_i + (j-1)T_i$
- C_i is constant.
- $d_{i,j} = \Phi_i + (j-1)T_i + D_i$
- The tasks are independent

4.2 Rate Monotonic

Always schedule the task that has the shortest period. Sufficient (but not necessary) schedulability test:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

4.3 Deadline Monotonic

Always schedule the task that has the shortest relative deadline. Sufficient (but not necessary) schedulability test:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

4.4 Necessary and sufficient schedulability test

We define the interference I_i for task i as

$$I_i(t) = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

where the tasks are ordered such that $m < n \iff D_m < D_n$

4.4.1 Algorithm

```
foreach (τ_i ∈ Γ) {
        I = 0;
        do {
                R = I + C_i;
                if (R > D_i) return false; // unschedulable
                I = sum_{j=1}^{i-1} ceil(R / T_j) C_j;
        } while (I + C_i > R);
}
return true;
```

4.5 EDF

Always schedule the task with the earliest deadline. Schedulable with EDF iff $\sum_{i=1}^{n} \frac{C_i}{T_i} = U \leq 1$.