CSC9006: Real-Time Systems

Real-Time Scheduling

Instructor: Chao Wang 王超

Department of Computer Science and Information Engineering

NATIONAL TAIWAN NORMAL UNIVERSITY

Course logistics

- Course website: https://wangc86.github.io/csc9006/
  - Homework submission: via NTNU Moodle (https://moodle.ntnu.edu.tw/)
- Course meetings: Thursdays 9:10-12:10 in S403, Gongguan Campus
- Instructor: Chao Wang 王超 (https://wangc86.github.io/)
  - Email: cw@ntnu.edu.tw
  - Office: Room 511, Applied Science Building, Gongguan Campus
  - Office hours: Tuesdays and Wednesdays, 9-11am

The busy world within a computing system

- We use our smart phones for a lot of tasks:
  - Writing e-mail/LINE
  - GPS navigation
  - Streaming YouTube videos
  - Booking a train ticket
  - …

… and we expect all of them are *responsive*

- How does a computing system meet this need?

Exercise: Timescale in timing constraints

- In our daily lives, what might be a reasonable response time, in microseconds/milliseconds/seconds/hours, for
  1. Moving your hero avatar in an online multiplayer game (e.g., LoL)
  2. Submitting your homework
  3. Stopping heating your bento if the microwave oven door is opened
  4. Analyzing telemetry data to create a weather forecast report
  5. Trading stock when you saw some breaking news (e.g., Google bought Apple, or vice versa)
  6. Streaming a live NBA game
  7. Using Hello Google/Siri

Real-time task model

- A task is a program that runs periodically
  - After each fixed period, the system releases a job of the task

    (For a one-time task, period = ∞)
- Execution time of a task is the amount of time it needs to run before completion (without obstruction)
- Response time of a task is the *observed* amount of time to complete (with delay caused by other tasks, etc.)
- Deadline of a task specifies a constraint for the response time
  - *Relative* deadline: defined according to application’s timing constraints
  - *Absolute* deadline: job release time + relative deadline

Real-world timing constraints

- *Hard* deadline: missing it will have serious consequence
  - Examples: tasks for a self-driving car or a pacemaker
- *Soft* deadline: missing it may cause some inconvenience
  - Examples: tasks for ________

- A cloud service needs to respond within 0.1 second to be considered responsive[1]

1 Jeff Dean et al. (Google) “The tail at scale.” Communications of the ACM 56.2 (2013)

Scheduling real-time tasks

- Given a set of tasks (i.e., a task set), a scheduling algorithm is a policy for a system to determine which job to execute first.

- A task set is schedulable if all jobs can meet their deadlines.

- A definition of an optimal scheduling algorithm:

  If a task set is unschedulable under an optimal algorithm, it is unschedulable under *any* other algorithms.

Optimal scheduling algorithms

- Rate Monotonic (**RM**) scheduling algorithm
  - Assign fixed priority levels to tasks in the order of their rates (i.e., 1 / period)
  - Optimal for the category of fixed-priority preemptive scheduling
- Earliest Deadline First (**EDF**) scheduling algorithm
  - Assign priority levels to tasks in the order of their *absolute* deadlines
  - Optimal for the category of dynamic-priority preemptive scheduling

Utilization bound – a simple schedulability test

- The utilization of a CPU core, U, is the percentage of its busy time
- Definition of a utilization bound U_b:
  - All tasks are guaranteed to be schedulable if $U \leq U_b$
- If a task set would lead to $U > 1$, then it implies that no scheduling algorithm can guarantee its schedulability

Comparison of RM and EDF algorithms

- RM
  - $U_b \leq 0.693$ [1]
    - may not guarantee schedulability even if the CPU core is not fully utilized.
  - Low overhead – no change of priority levels at run-time

- EDF
  - $U_b \leq 1$
    - Guarantees schedulability as long as the CPU core is not overly utilized.
  - Relative higher overhead

1 Liu, Chung Laung, and James W. Layland. "Scheduling algorithms for multiprogramming in a hard-real-time environment." *Journal of the ACM (JACM)* 20.1 (1973): 46-61.

Assumptions

- So far, we assume the following:
  - Single processor (using only one CPU core)
  - All tasks are periodic
  - Relative deadline = period (called the *implicit deadline*)
  - No priority inversion.

Further reading

- Davis, Robert I., and Alan Burns. "A survey of hard real-time scheduling for multiprocessor systems." *ACM computing surveys (CSUR)* 43.4 (2011): 1-44.