CSC9006: Real-Time Systems

Real-Time Scheduling (2)

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

Outline of today's lecture

- More on schedulability test

- Scheduling dependent tasks
  - Priority ceiling protocol
  - Priority inheritance protocol

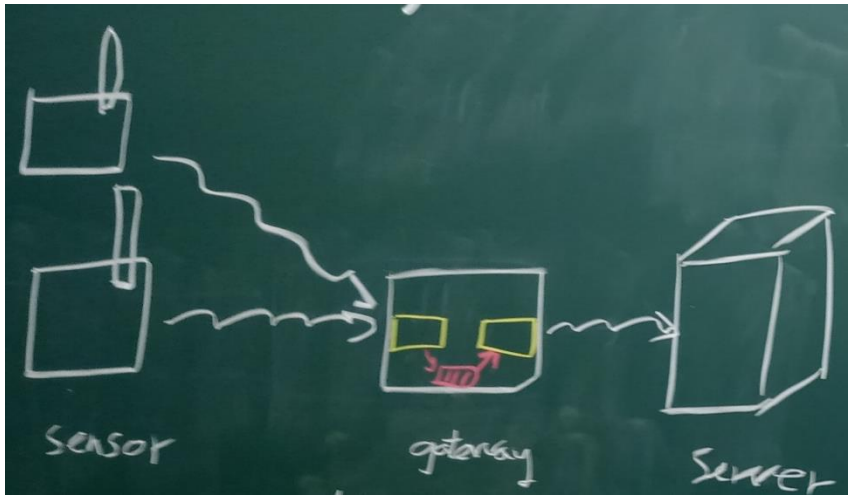
Types of schedulability test

- Given a task set, there are three types of schedulability test
  - Sufficient test: The test said it is schedulable, and in fact it is.
  - Necessary test: The test said it is not schedulable, and in fact it is not.
  - Exact test: The test that is both sufficient and necessary.
- Using the utilization bound as a scheduling test
  - $U_b \leq$ (the # of CPU cores) is necessary for any scheduling algorithm
  - $U_b \leq 1$ for EDF is exact for single-core case
  - $U_b \leq \ln 2 = 0.693$ for RM is sufficient for single-core case (but not necessary)
    - E.g., a task set having only one single task, with utilization=0.8

Scheduling dependent tasks

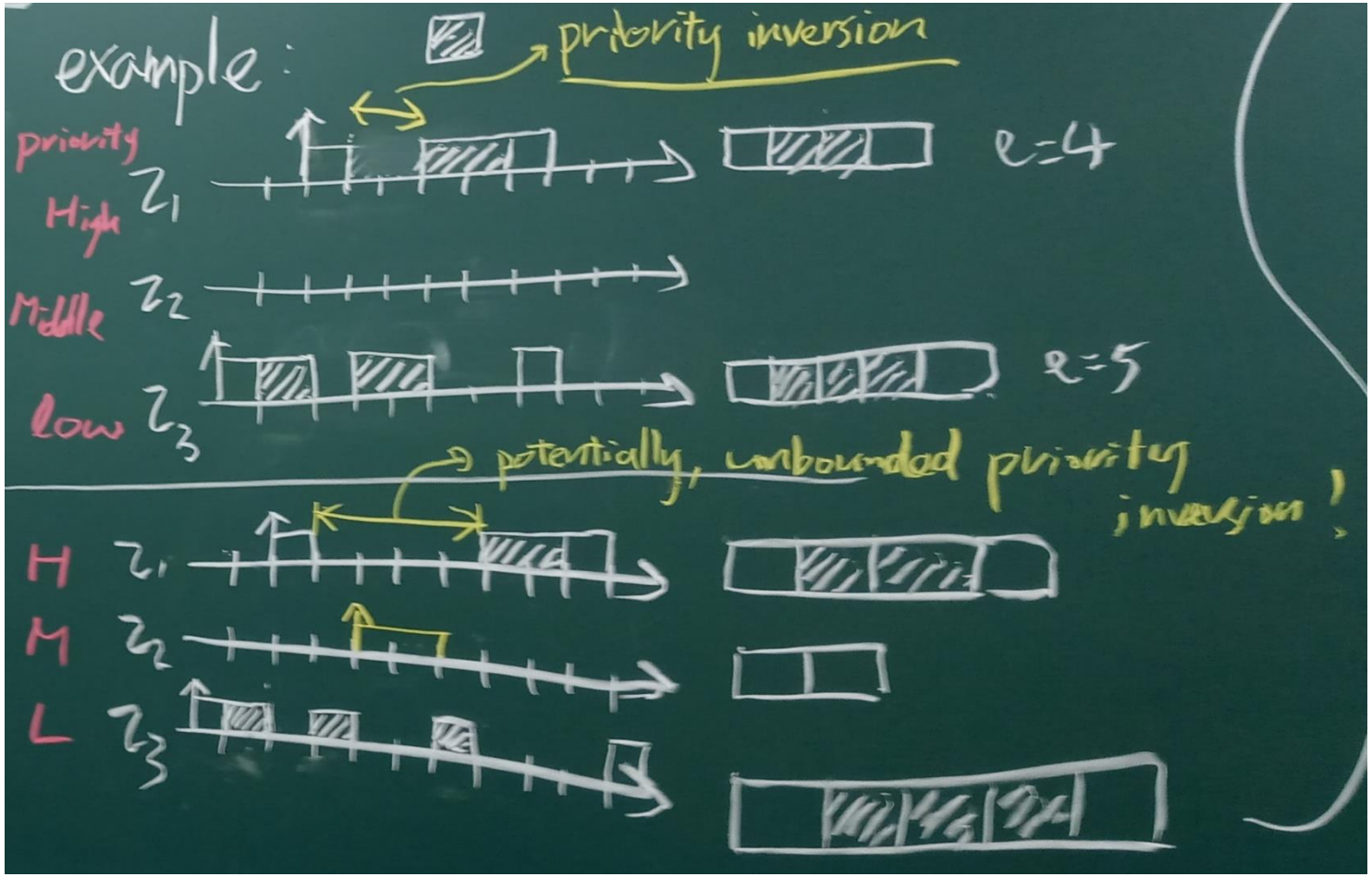
- In many cases, tasks in a system depends on each other
  - For example, in an IoT gateway, a receiving task passes data to a sending task via a shared memory
    - In this case, a *semaphore* is used to prevent concurrent executions of the same section of code. The semaphore in this case is also known as a *mutex*.
    - The tasks waiting for the mutex are delayed.

Dependent tasks in an IoT gateway:



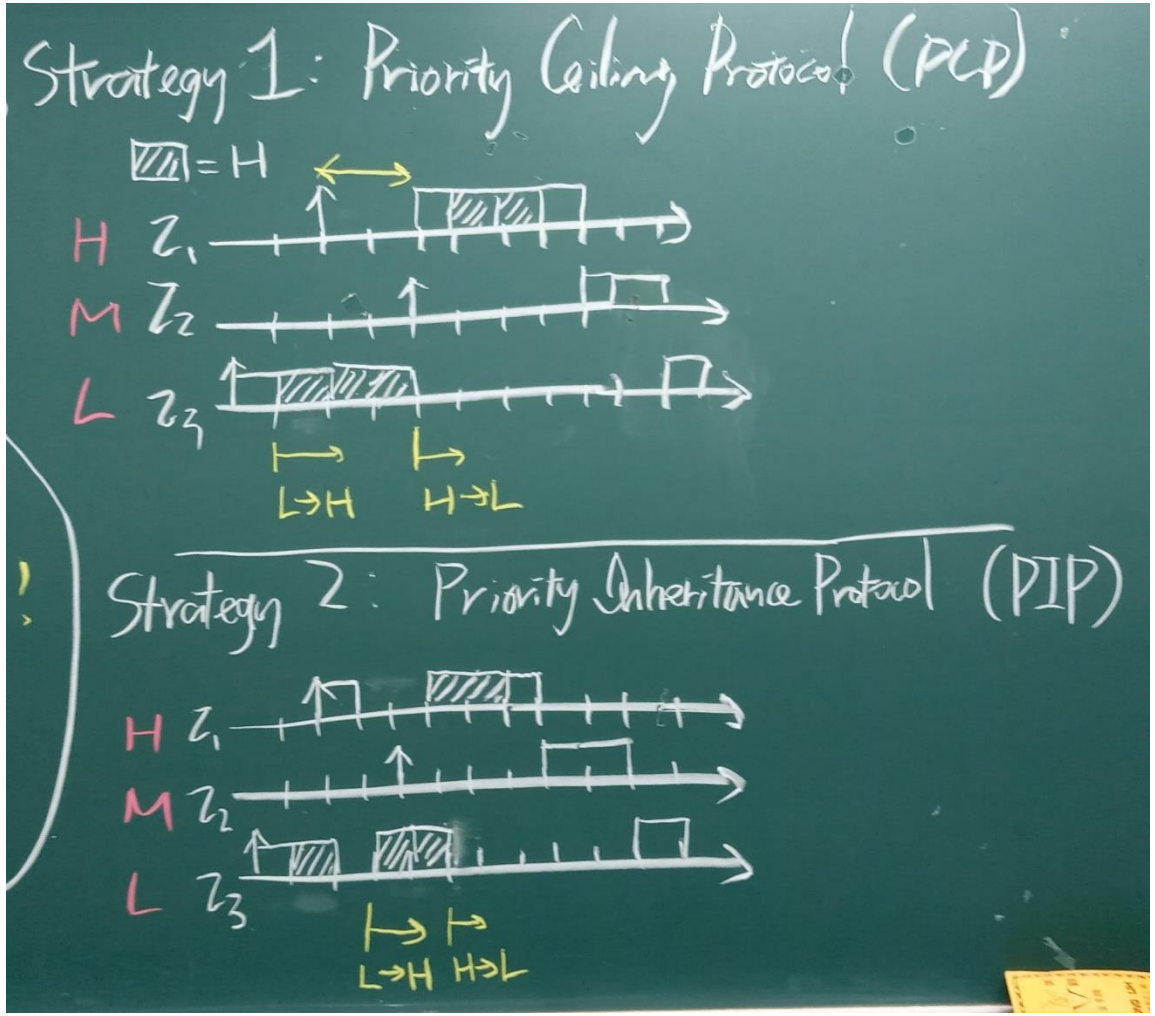
Priority inversion

- A high-priority task being blocked by some low priority tasks.
- In some cases, priority inversion may greatly delay the high-priority tasks



Mitigating the delay caused by priority inversion

- Priority Ceiling Protocol: Assign a critical section with the priority level equal to that of the highest-priority task that may access it. A task entering the critical section has its priority level raised to the priority level of the critical section.
- Priority Inheritance Protocol: The priority level of a task is raised to that of the highest-priority task *currently* being blocked at the critical section.



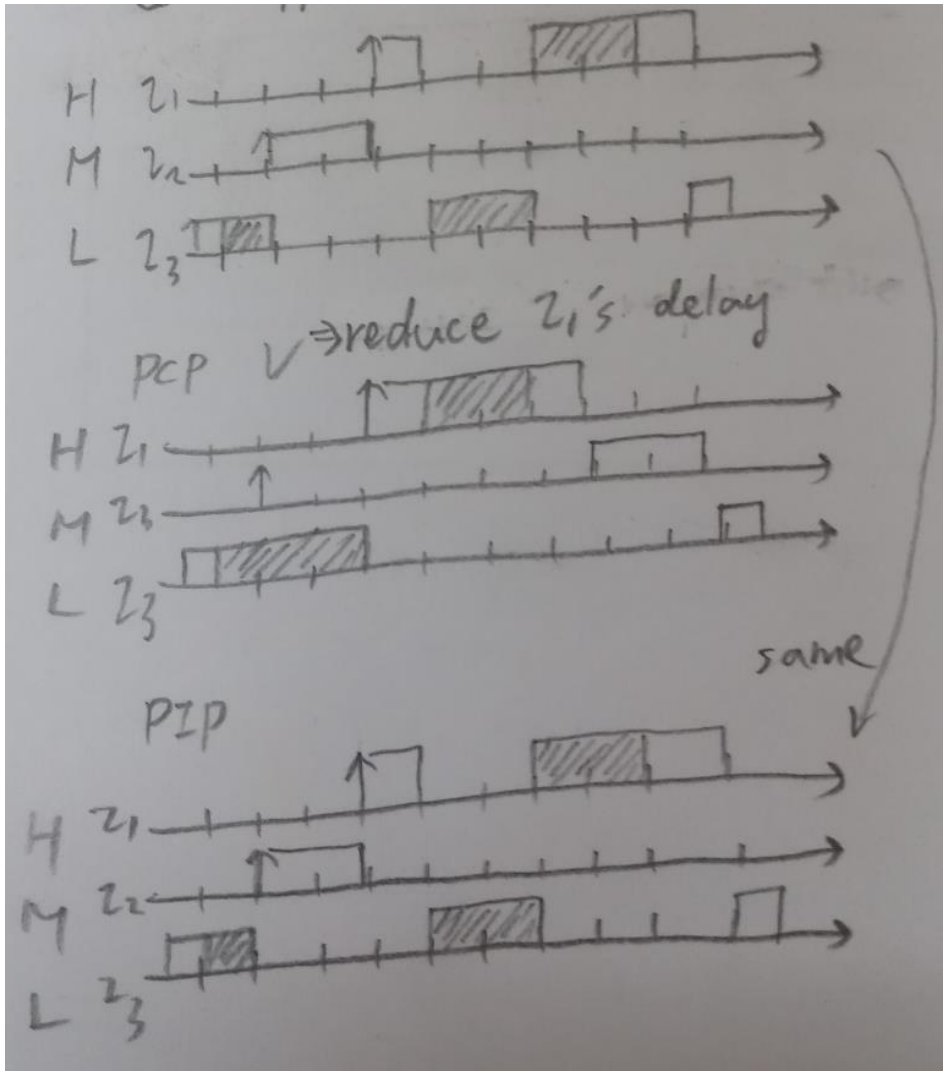
Comparing PCP and PIP

- PCP and PIP each has its own benefits (can you name one for each?)

- Now let's study three example scenarios to see how they work and how they differ from each other!

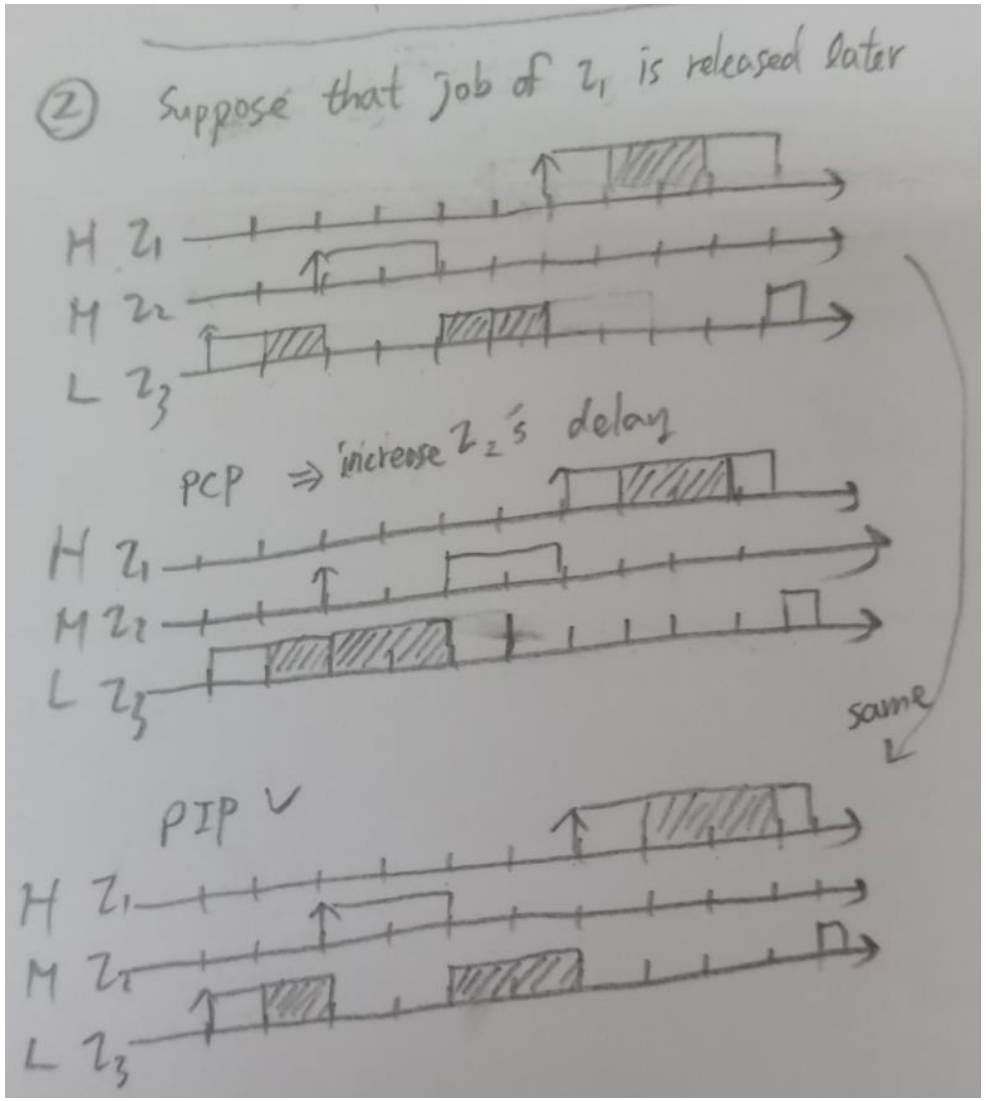
Scenario 1

- Job of τ_1 released at t_1=4
- Job of τ_2 released at t_2=2
- Job of τ_3 released at t_3=0



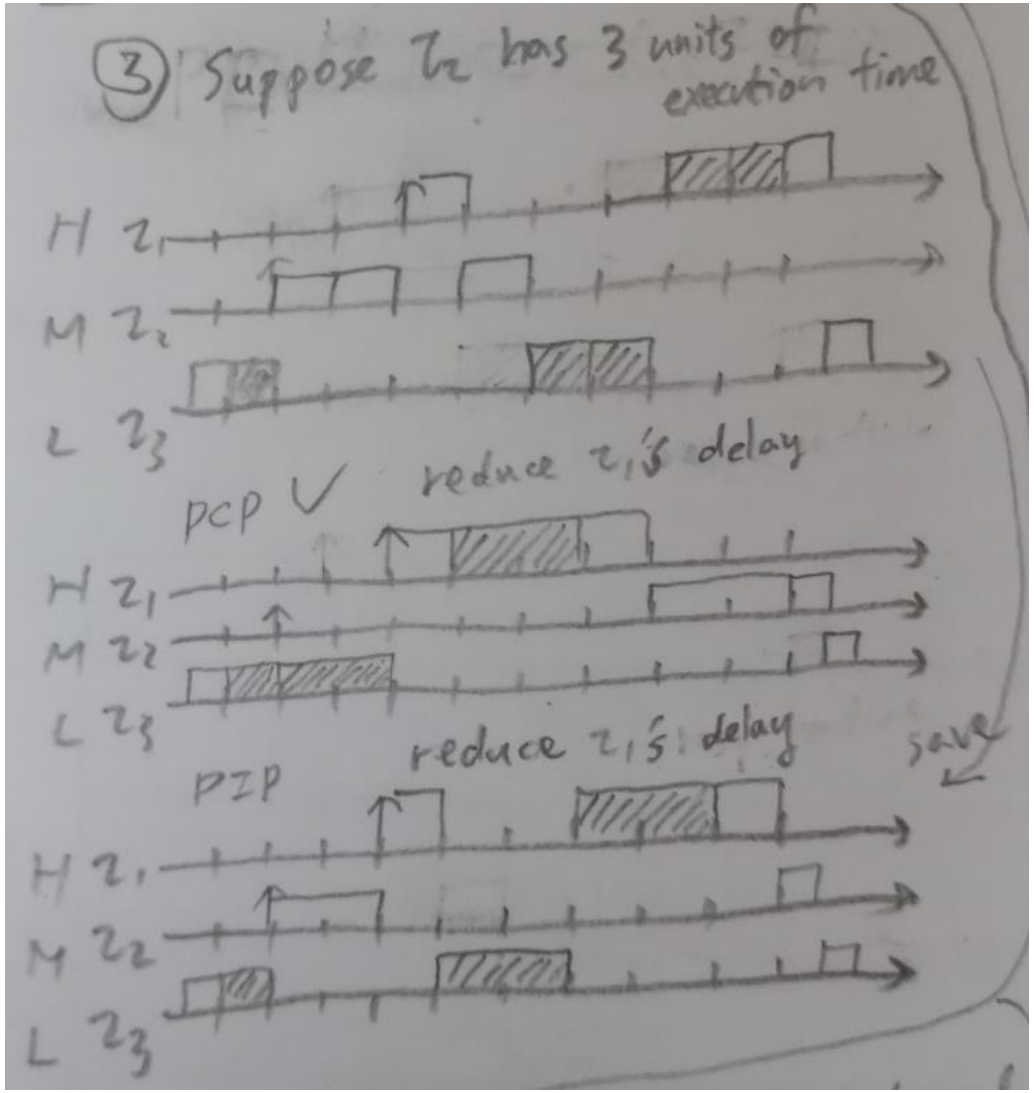
Scenario 2

- Job of τ_1 released at t_1=6
- Job of τ_2 released at t_2=2
- Job of τ_3 released at t_3=0



Scenario 3

- Suppose τ_3 has 3 units of execution time
- Job of τ_1 released at t_1=4
- Job of τ_2 released at t_2=2
- Job of τ_3 released at t_3=0



Lessons from the three scenarios

- PCP aims to have the task in a critical section finish sooner
- PIP aims to allow middle-priority tasks execute normally, as it raises the priority level of the low-priority task only when needed
- Both PCP and PIP may reduce undesirable priority inversion
- If there's no undesirable priority inversion in the first place,
  - PCP may or *may not* perform better than doing nothing
  - PIP may introduce no effect

Further reading

- “What really happened on Mars Rover Pathfinder.”
  - A wide-spread summary: http://catless.ncl.ac.uk/Risks/19.49.html#subj1
  - The account from the software team for the Mars Pathfinder spacecraft: https://www.cs.unc.edu/~anderson/teach/comp790/papers/mars_pathfinder_long_version.html
  - A more recent retold of the story: https://www.rapitasystems.com/blog/what-really-happened-to-the-software-on-the-mars-pathfinder-spacecraft
- 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.