Scheduling in Real-Time Systems

- Dynamic: Schedules depend on run-time situation.
  - More flexible, more efficient (most soft real-time systems).

Scheduling: Earliest Deadline First (EDF)

1. Determine (non-empty) the process set with the earliest deadline.
2. Execute this process
   - ... until another process' deadline is found earlier than the current one.

Real-time scheduling: Earliest Deadline First

Times to deadlines

Worst case response times

Maximal utilization

Fixed Priority Scheduling (FPS), rate monotonic

1. At run-time: dispatch the runnable process with the highest priority.

Deployment

A scheduling scheme provides two features:

- Ordering: the use of resources (e.g., CPUs, networks) in a live system may make it harder to predict the timing behaviour.
- Non-determinism: Non-determinism may make it harder to predict the worst-case behaviour of the system when the scheduling algorithm is applied.

Some assumptions:

- All processes are independent.
- Pre-emptive or cycle done.
- All processes are fixed and all periods are fixed.
- The task-switching overhead is negligible.

A simple process model

- The number of processors in the system is fixed.
- All processors are periodic and all periods are known.
- All processors are independent.
- The task-switching overhead is negligible.
- All deadlines are identical with the process cycle times (periods).
- The worst-case execution time is known for all processes.
- All processors are released at once.

Consequences:

- In the context of fixed priority schedulers (i.e., processes in the framework of fixed priority systems, it means in this model or in the diachrony in the chapter):
  - Deadlines, constraints, other means are needed to select a process, i.e., a pre-emptive scheme.
  - Dynamic scheme: the specified process is selected at run-time, due to the current deadline.

Gives an idea how "critical" the schedule is.

In the example: Worst case response times are identical to cycle times.
Real-time scheduling: Fixed Priority Scheduling

Task set

Execute FPS schedule vs Fails!

Reduced task set

Execute FPS schedule vs Works!

Further reduced task set

Execute FPS schedule vs Works!

Worst case response times

Response time analysis

The worst case for Earliest Deadline First is not necessary when all tasks are released at once!

for others tasks: \( R_j = C_j = \sum_{i=0}^{n-1} C_i \) (interference from higher priority tasks)

for others tasks: \( R_j = C_j = \sum_{i=0}^{n-1} \frac{R_i}{D_i} C_i \)
Scheduling

Real-time scheduling: Earliest Deadline First

Response time analysis

\[ R_j(a) = \frac{1}{C_j} + \sum_{i=1}^{n-1} \left( \frac{a - T_j + T_i}{C_j} \right) + C_j \]

Worst case response times

\[ R_j(a) = \frac{1}{C_j} + \sum_{i=1}^{n-1} \left( \frac{a - T_j + T_i}{C_j} \right) + C_j \]

Scheduling

Real-time scheduling: Fixed Priority Scheduling

Worst case response times

\[ R_j(a) = \frac{1}{C_j} + \sum_{i=1}^{n-1} \left( \frac{a - T_j + T_i}{C_j} \right) + C_j \]

Testing all combinations in a hyper-cycle

\[ R_j = \max \left( \frac{a}{C_j} \right) \]

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known

Scheduling

Real-world extension: More realistic assumptions

- Tasks are periodic
- Deadlines are identical with task's period time \((D = T)\)
- Tasks are independent
- Pre-emptive scheduling
- Worst case execution times are known
Scheduling

Sporadic and Aperiodic Processes

FP S for hard real-time tasks

Sporadic and Aperiodic Processes

Introducing a soft real-time task

Sporadic and Aperiodic Processes

FP S lowest priority for soft real-time task

Sporadic and Aperiodic Processes

Introducing a server task on highest priority

Sporadic and Aperiodic Processes

FP S task as normal task

Sporadic and Aperiodic Processes

Sporadic task utilizing deferrable server

Sporadic and Aperiodic Processes

Sporadic task utilizing sporadic server

Sporadic and Aperiodic Processes

FP S with dual priorities

Scheduling

Tasks with arbitrary deadlines

Tasks with D < T (Deadline under their own control)

FP S for hard real-time tasks

FP S

Sporadic task utilizing IDP server

Sporadic and Aperiodic Processes

Earliest Deadline Last (EDL) for sporadic tasks

Introducing an EDF server

FP S

Sporadic task utilizing sporadic server

Sporadic and Aperiodic Processes

Real World M Extension

More Realistic Assumptions

DMPO

DMPO

DMPO

DMPO

DMPO

DMPO

DMPO
Tasks with $D > T$

\[ R_j' \text{ for a specific scheduling situation, } \]

\[ T \text{ being a boolean function returning } \]

\[ D \text{ for } R_j' \text{ which violate DMPO:} \]

\[ W' \]

\[ G \]

\[ \text{Swap all } t_j \text{ in } Q, \text{ with } W > T_i \text{ resulting in all } t_j, \]

\[ D > T_i \text{ is released once in } \]

\[ R_i \] is released once in \]

\[ L_1 \]

\[ L_3 \]

\[ D \text{ for } R_j' \text{ which violate DMPO:} \]

\[ W' \]

\[ G \]

\[ \text{Swap two priorities out of } W \text{ which violate DMPO:} \]

\[ \text{Tasks with } D > T \]

\[ \text{Tasks are } \]

\[ \text{Pre-emptive scheduling} \]

\[ \text{Worst case computation time in critical section} \]

\[ \text{Proof of DMPO optimality} \]

\[ \text{Swap two priorities out of } W \text{ which violate DMPO:} \]

\[ \text{Tasks with } D > T \]

\[ \text{Tasks are } \]

\[ \text{Pre-emptive scheduling} \]

\[ \text{Worst case computation time in critical section} \]

\[ \text{Proof of DMPO optimality} \]

\[ \text{Swap two priorities out of } W \text{ which violate DMPO:} \]

\[ \text{Tasks with } D > T \]

\[ \text{Tasks are } \]

\[ \text{Pre-emptive scheduling} \]

\[ \text{Worst case computation time in critical section} \]

\[ \text{Proof of DMPO optimality} \]
Interdependent tasks

- Task dependencies with multiple locks

Immediate ceiling priority protocol (POSIX, Ada, RT-Java)

- Each task $t_i$ has static priority $P_i$
- Each resource $R_j$ has static ceiling priority $C_j$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task $t_i$ has static priority $P_i$
- Resource $R_j$ has static ceiling priority $C_j$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$

Immediate ceiling priority protocol $POSIX, Ada, RT-Java$

- Task's period time $T_i$
- Task's release time $C_i$
Language support

Ada provides:

- Task and interrupt priorities (static, dynamic, active).
- Task attributes.
- Prioritized entry queues.
- Priority ceiling locking (ICPP).
- Schedulers (FPS with FIFO within priorities (pre-emptive), Round Robin, EDF).
- Task execution time measurements.

Ada does currently not provide:

- Sporadic servers.

POSIX provides:

- Threads and interrupt priorities (static, dynamic, active).
- Threads can be 'system contented' or 'process contented' (priority scheduling unclear in this case).
- Prioritized message queues.
- Priority ceiling locking (ICPP).
- Schedulers, priority based with at least:
  - FIFO, Round-Robin, Sporadic Server, possibly others.
  - Timers.

Language support

- Ada, POSIX

Summary

Basic real-time scheduling:
- Fixed Priority Scheduling (FPS) with Rate Monotonic (RMPO) and Deadline Monotonic Priority Ordering (DMPO).
- Earliest Deadline First (EDF).

Real-world extensions:
- Aperiodic, sporadic, soft real-time tasks.
- Deadlines different from period.
- Synchronized talks (priority inheritance, priority ceiling protocols).
- Noncooperative and deferred pre-emption scheduling.
- Fault tolerance in terms of exception handling considerations.

Language support:

- Ada, POSIX