Scheduling

Motivation

- especially for behaviors
  - pseudo-parallelism
  - i.e., processes share one (few) CPU(s) to emulate concurrent execution
- but also relevant for “higher” layers
  - modeling, planning, reasoning, etc. can profit from concurrency
- note: due to use of “normal” computers
  - Grey Walter’s tortoises (late 40s - 50s)
  - analog circuits (radio vacuum tubes)
  - differential drive, light sensors, bumper
  - homing in on white light => charging station
  - inherently parallel computations

Grey Walter’s Tortoises

Scheduling

- purpose
  - making use of a scarce resouce
  - typically processing power
  - but also harddisk, printer, etc. access
- time sharing aka time slicing
  - **time slot** aka **quantum**: constrained time interval for execution
  - **context switch**: change from one executed process to next one

Terminology

- task / process
  - "large", application
  - "independent", especially own address space
- thread
  - "small", part of a program (i.e., task/process)
  - shared address space

- no fixed terminology
  - depending e.g. on type of OS
- here:
  - process as term for the general category
  - task and thread as subclasses

Important Process Constraints

- deadline
  - a moment in time until when
  - the execution of the process must be finished
- ready time
  - earliest possible moment in time
  - where the process may be started
- period
  - minimal frequency at which a time slot
  - for a cyclic process must be provided
  - i.e., regular deadline

Realtime Processing

- hard real-time
  - fullfill all contraints at all times
  - e.g., control of critical components
  - often computed offline
- soft real-time
  - best effort
  - e.g., multimedia streams

- implementation of hard realtime
  - determination of max. runtime of a process
  - problems with normal OS/hardware
  - especially pipelining, cacheing, interrupts

Two major approaches

- cooperative scheduling
  - running process stops by itself
  - and invokes the scheduler
- preemptive scheduling
  - scheduler stops (preempts) running process
  - higher demands on the context switch

Finding an optimal schedule

- naive algorithm
  - test all possibilities
  - exponential runtime

$$n! \approx (\frac{n}{e})^n \cdot \sqrt{2\pi n}$$

- is a better algorithm possible?
  - n non-preemptive processes
  - with ready-times, execution times, deadline
  - searching optimal schedule is NP-complete
  - can be reduced to bin-packing

All processes created equal, or not?

- round robin scheduling
  - most simple and very commonly used
  - FIFO principle
- but
  - not all processes of same "importance"
  - need for "preferences"

  => use of priorities

B-Scheduling

Behaviors: Need of Scheduling

- several behaviors “active” at the same time
  - pseudo-parallelism
  - need to decide which process runs next
- simple solution
  - run one after the other
  - i.e., round robin scheduling

Problems RR behavior scheduling

- many behaviors at different time scales
  – e.g., motor control vs battery monitor
  – proper scheduling with priorities needed
- behaviors are used for control
  – not only deadlines
  – but also minimization of jitter,
  – i.e., variations in the execution period

Handling different Time-Scales: Exponential Effect Priorities

exponential effect:

priority incremented

=> frequency halved

- process *P[i]*
- priority *prio[i]*
  - *prio[x] == 0* :
    maximum frequency
  - *prio[x] == n* :
    1/2 frequency of n-1

example

priority	frequency
0	1024 Hz
1	512 Hz
2	256 Hz
3	128 Hz
4	64 Hz
...	...
10	1 Hz
11	every 2 sec
...	...
16	~1 minute
...	...
22	~1.1 hour