Linux Scheduling

Scheduling Policy and Algorithms, the schedule() Function
of the Linux Kernel version 2.4.20

Patrick Stahlberg
<Patrick.Stahlberg@hadiko.de>

$Id: speech.mgp,v 1.7 2002/12/17 10:59:17 patrick Exp $

Structure of this talk

- What is scheduling, why do we need it?
  - concepts related to scheduling
- How is scheduling done in Linux?
  - policy
  - algorithms

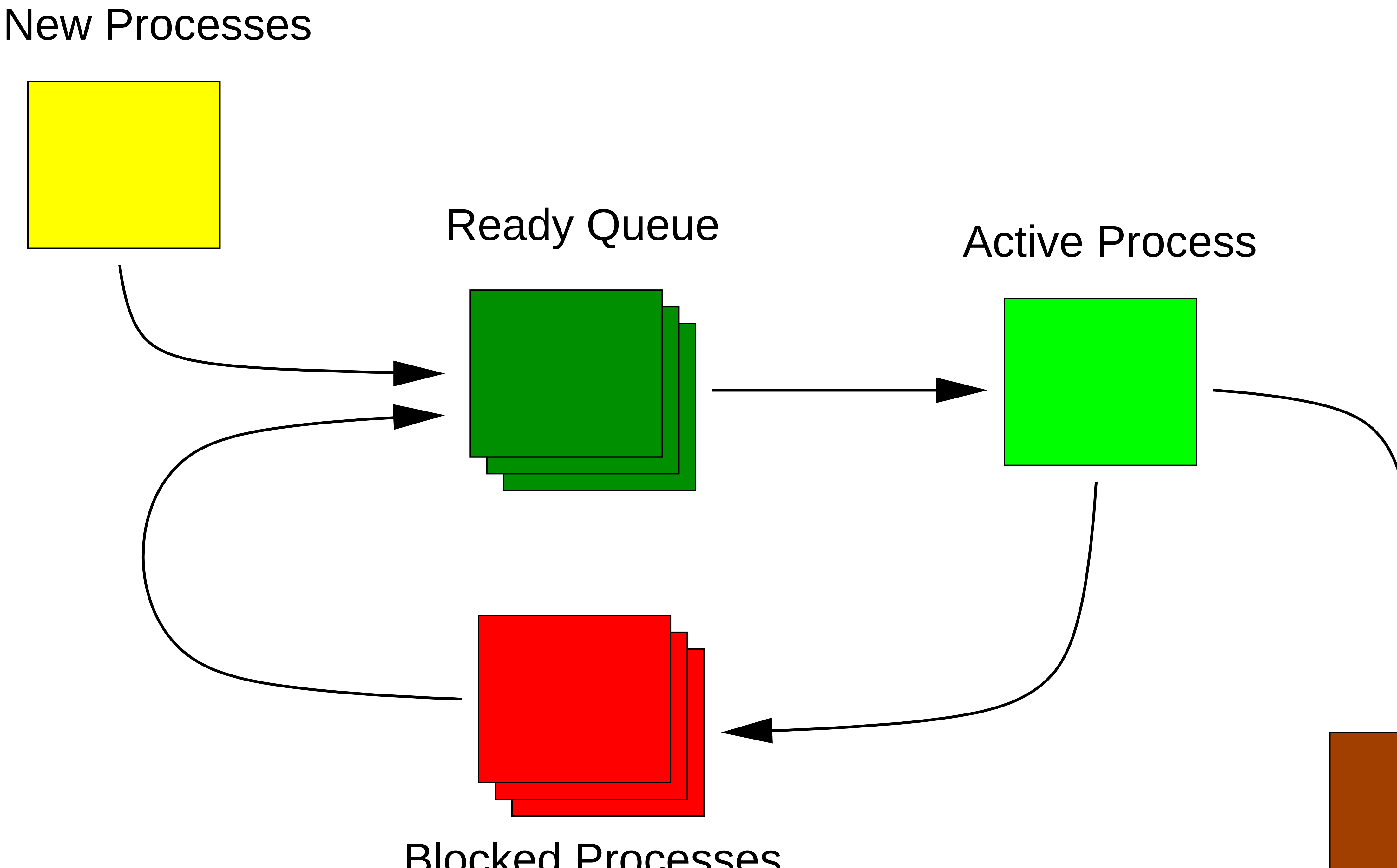
Part One

What is scheduling?

What is scheduling?

- Distribution of the resource ’processor’ to the competing tasks
- In this talk: only uniprocessor-scheduling

Lifecycle of a process



Classification of processes

- Interactive processes
- Batch processes
- Real-time processes

- I/O-bound
- CPU-bound

- These classifications are independent

Process Preemption

- Ability of an OS to take away CPU control of a process before it does this voluntarily.
- Processes are assigned processing time quanta, a process will be preempted when its quantum duration is passed.
- Scenario: a high-priority task enters the TASK_RUNNING state while a low-priority task is active --> the low-priority task is preempted
- Linux features preemptive processes but not (yet) a preemptive kernel

Measures of good Scheduling (1)

- Fairness, equal treatment of processes
- Prevent "Starvation" of processes
- Use processing power efficiently
- Minimize overhead caused by scheduling itself

Measures of good Scheduling (2)

- For a Multiuser-Multitasking-OS:
  - Interactive processes should have quick response times
  - Desirable: intelligent treatment of I/O- and CPU-bound processes

Part Two

Linux scheduling policy and algorithms

When is the scheduler called?

- Direct invocation
  - During System Calls
  - Mostly indirectly via sleep_on()
  - e.g. when waiting for a resource

- Lazy invocation
  - After System Calls or interrupts
  - if need_resched is set
  - e.g. after sched_set_scheduler()
  - The timer interupt also sets need_resched, making sure that schedule() is called frequently

Data structures used by the scheduler

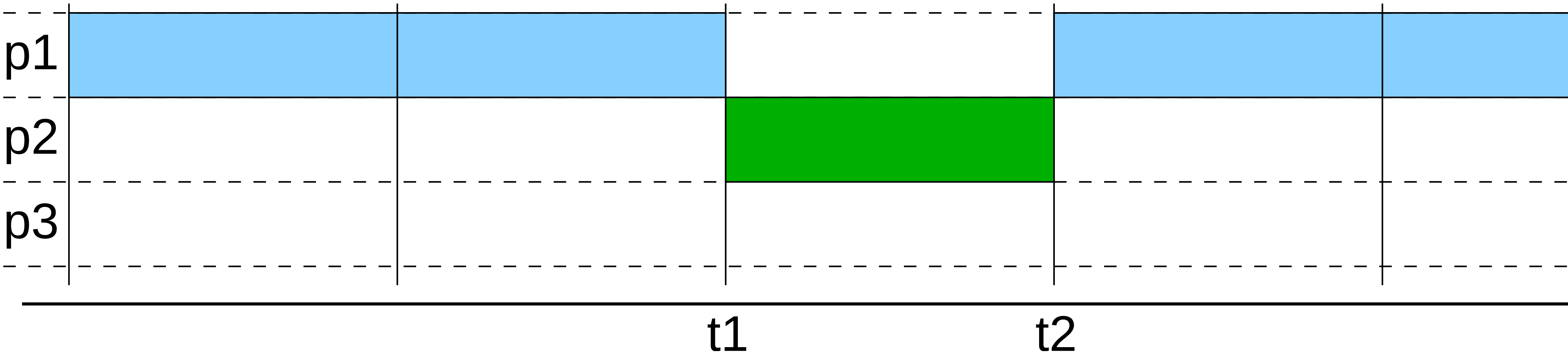
- need_resched
  - A flag set by interrupt handling routines
  - When set, ret_from_intr() calls schedule()
- policy
  - Scheduling policy, see following slide
- rt_priority
  - Static priority field for real-time processses
- priority
  - Base time quantum (SCHED_RR)
  - Base priority (SCHED_OTHER)
- counter
  - CPU time left for process in current epoch

Scheduling classes

- Linux provides three different scheduling algorithms called ‘scheduling classes’
- Each process can be assigned one scheduling class
- Scheduling classes are: SCHED_FIFO, SCHED_RR, SCHED_OTHER

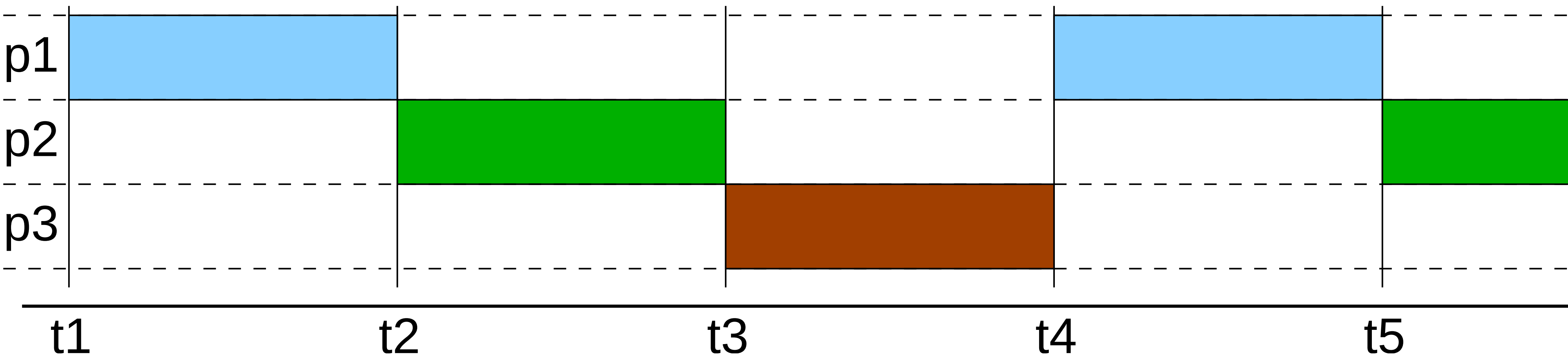
The SCHED_FIFO scheduling class

- Real-time processes
- Unlimited CPU time for processes given that there is no higher-priority process
- Static priority



The SCHED_RR scheduling class

- Real-time processes
- Enhancement of SCHED_FIFO that introduces time slicing
- Static priority



The SCHED_OTHER scheduling class

- All other processes
- Dynamic priority
- Time slicing
- Time slicing is using epochs

Epochs

- Each non-realtime process is assigned a time quantum at the beginning of an epoch.
- The epoch ends when all processes in the runqueue have used up their time quantum.

The schedule() function

Very much simplified:

- If previous process is a SCHED_RR process which has exhausted its time slice: assigns a new time slice to it and puts it at the end of runqueue.

- Main scheduling loop:
  - Loops through items of runqueue
  - Calculates a ‘goodness’ value for each one of them
  - Remembers the first task with highest goodness value

- Does a context switch to the chosen task.

Goodness of a process

- Calculated by the goodness() function
- Goodness of real-time tasks is always higher than goodness of a SCHED_OTHER task (1000 + rt_priority)
- Goodness is calculated like this for SCHED_OTHER tasks:

```
if (p->mm == prev->mm)
    return p->counter + 1 + 20 - p->nice;
else
    return p->counter + 20 - p->nice;
```

Literature:

- kernel/sched.c
- http://en.tldp.org/LDP/tlk/kernel/processes.html#tth_sEc4.3
- sched_setscheduler(2)
- http://www.kernelnewbies.org/documents/schedule/
- http://www.ora.com/catalog/linuxkernel/chapter/ch10.html

End. Questions?