Scuola Superiore Sant'Anna di Studi Universitari e di Perfezionamento

Research on Soft Real-time and Virtualised Applications on Linux

Tommaso Cucinotta

Real-Time Systems Lab (RETIS)

Center for Excellence in Information, Communication and Perception Engineering (CEIICP)

Scuola Superiore Sant'Anna, Pisa (Italy)

Workshop Outline

Modules of the workshop

- Motivations and Background
- Real-Time Scheduling on GPOSes and Linux
- Improving Linux for Real-Time Applications
- Optimum Deployment of Virtual Machines
- Probabilistic Real-Time Guarantees
- Adaptive Real-Time Scheduling
- New Research Themes

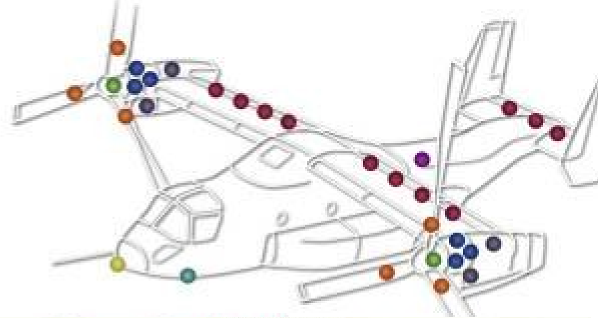
Motivations and background

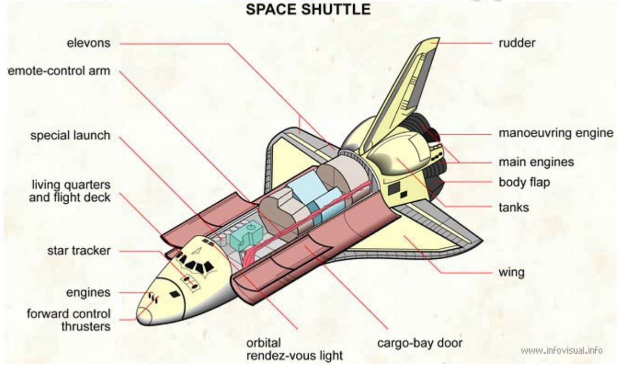
What is Real-Time

Drive assistance

- Engine control, brakes, stability, speed, parking
- Trajectory and set-up control

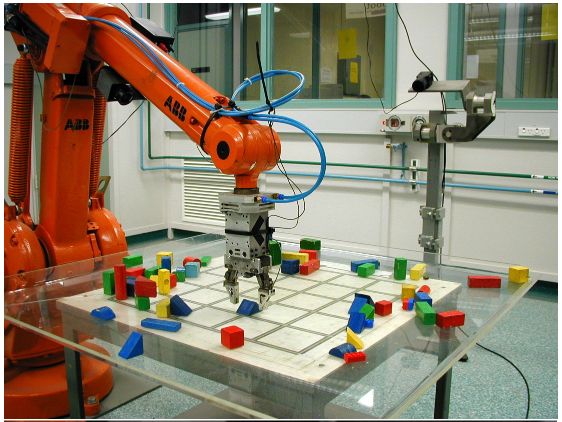
Defence, army, space

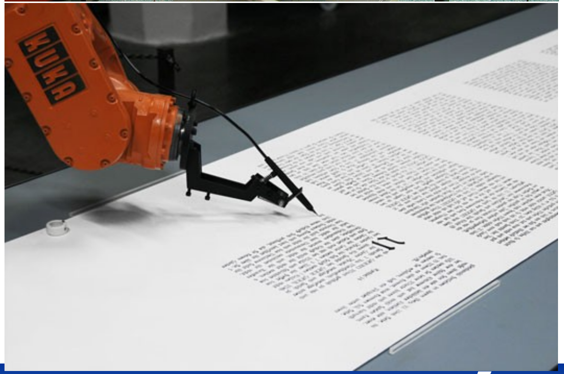


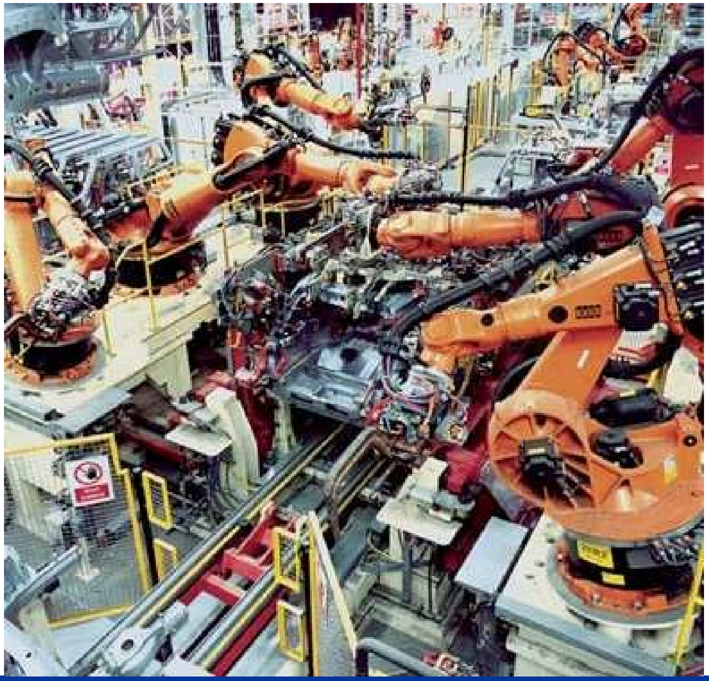


What is Real-Time

- **Control of chemical and nuclear plants**
- **Control of productive processes and industrial automation**
- **Traffic control**

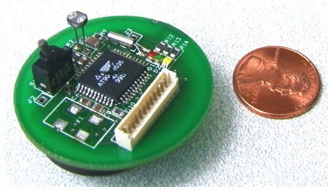


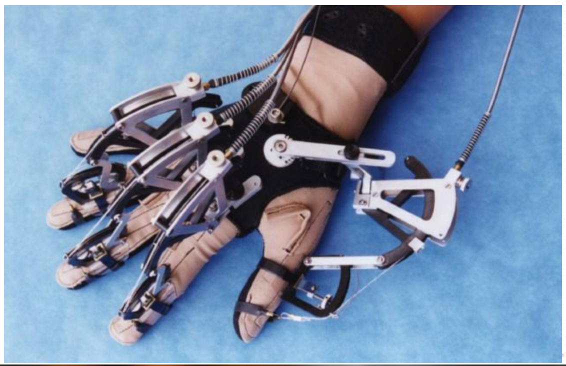


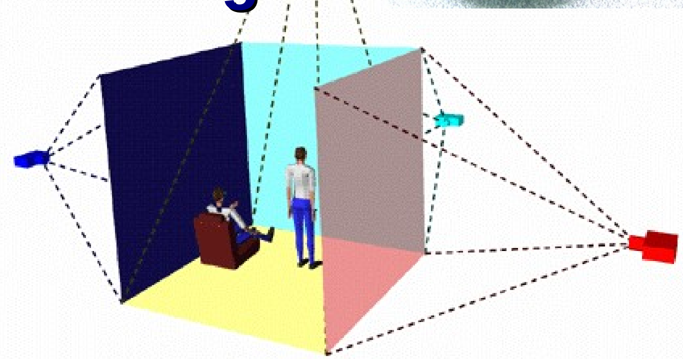


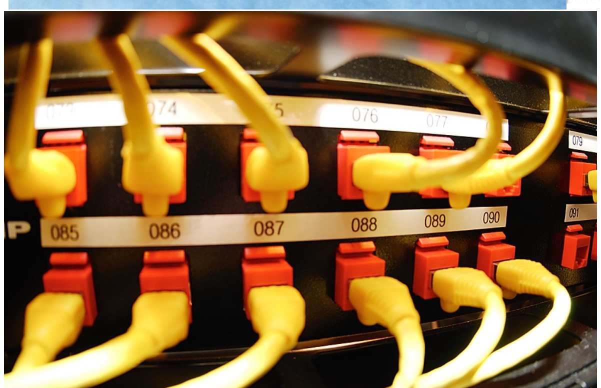
What is Real-Time

- **Multimedia, videosurveillance**
- **Augmented virtual reality**
- **Telecommunications**
- **Environment monitoring**









Criticality of time requirements

Systems with critical timing requirements

- e.g., defence, army, space

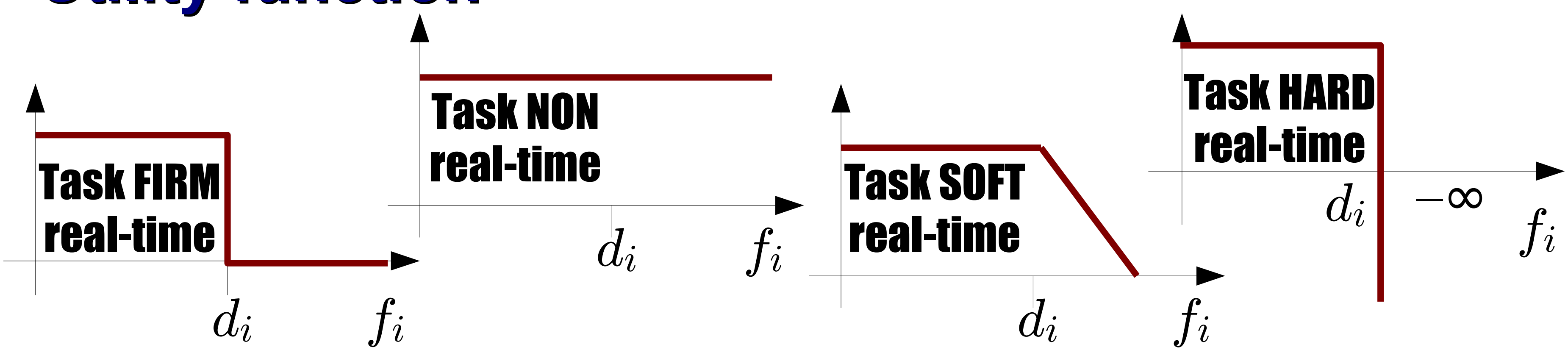
Systems with lower criticality timing requirements

- e.g., industrial automation

Systems with non-critical timing requirements

- e.g., multimedia, virtual reality, telecommunications

Utility function



Our Focus

We focus on systems

- With **non-critical soft real-time** requirements
- Where the use of a **GPOS is desirable and feasible**
  - As opposed to a **traditional RTOS**

Motivations

General-Purpose Operating Systems

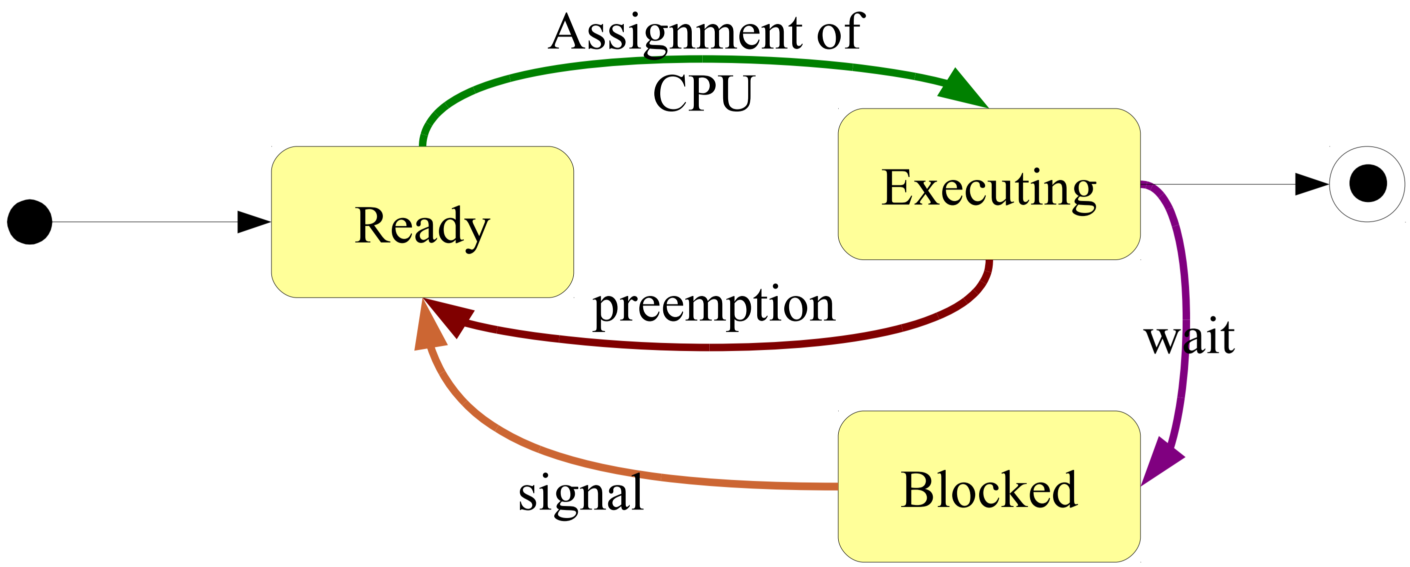
- Very effective for storing & managing multimedia contents
- Designed for
  - **average-case** performance
  - serving applications on a **best-effort** basis
- They are not the best candidate for serving *real-time applications* with **tight timing constraints**
  - nor for **real-time multimedia**

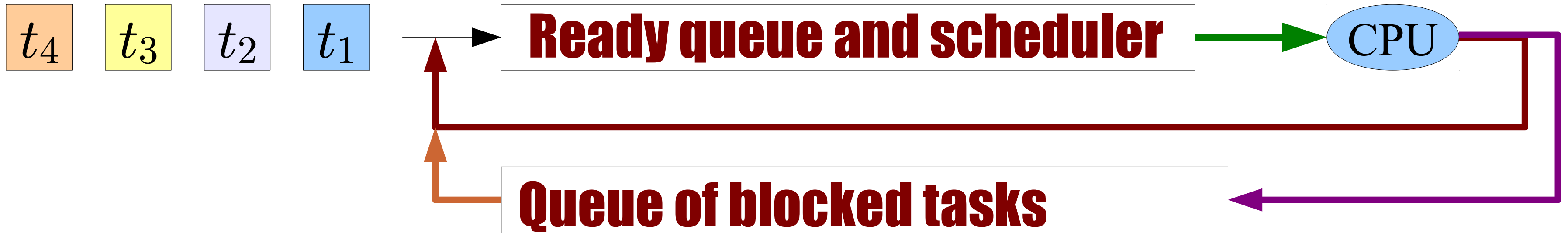
Motivations

Overcoming limitations of a GPOS for multimedia

- **Large buffers** used to compensate *unpredictability*
  - ==> poor real-time interactivity and no low-latency multimedia
- **One-application one-system** paradigm
  - For example, for low-latency real-time audio processing (jack), gaming, CD/DVD burning, plant control, etc...
- **POSIX real-time extensions**
  - Priority-based, **no temporal isolation**
  - Not appropriate for deploying the multitude of (soft) real-time applications populating the systems of tomorrow

States of a process/thread/task

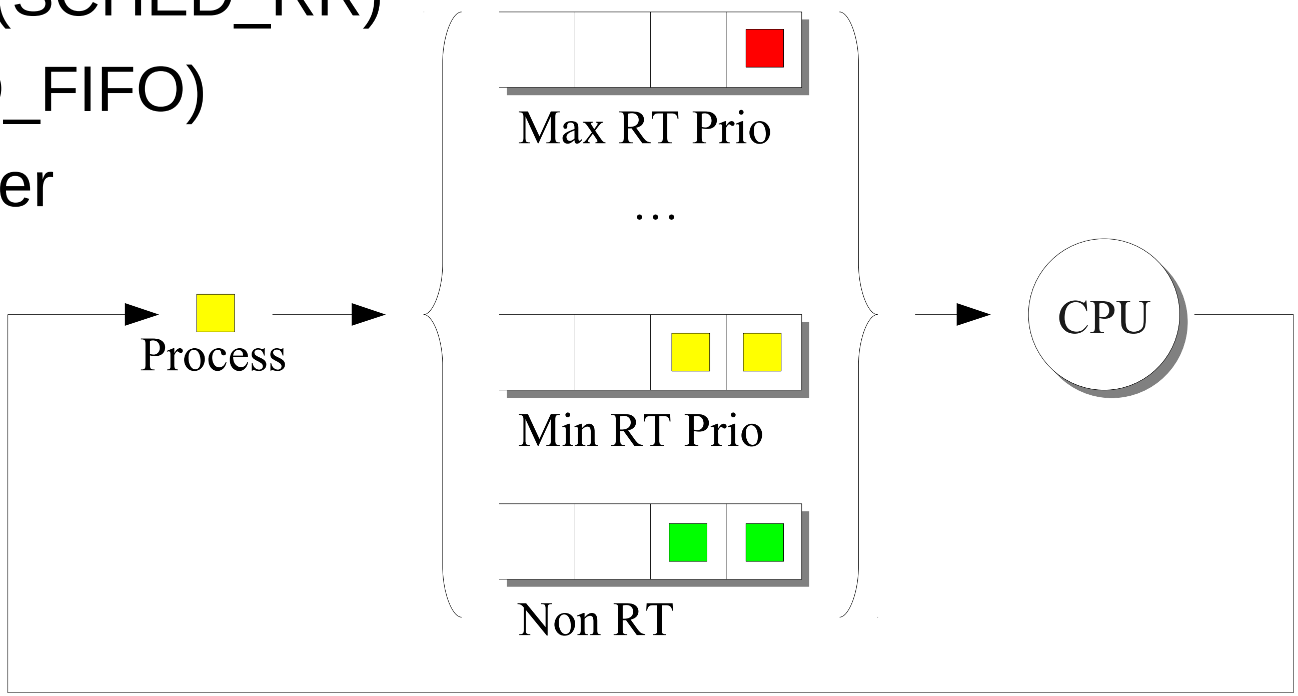




Multi-queue priority-based scheduler

Processes at same priority

- Round-Robin (SCHED_RR)
- FIFO (SCHED_FIFO)
- Sporadic Server
  (see later)



Traditional RT Systems (and Priority Scheduling)

All deadlines respected as far as system behaves as foreseen at design time

- Traditional (C, T) task model
  - C: Worst-Case Execution Time (WCET)
  - T: Minimum inter-arrival period

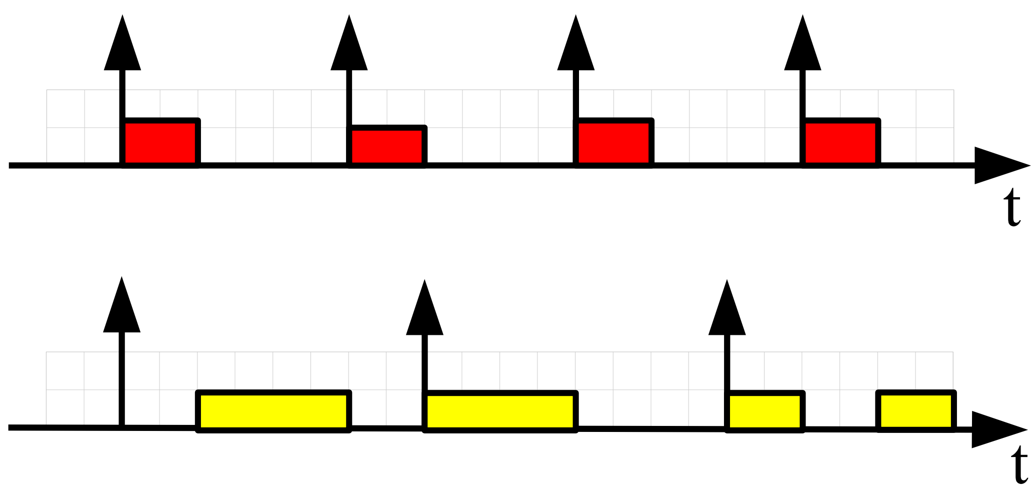
Admission Control, e.g., for RM:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(\sqrt[n]{2} - 1\right)$$

$$\prod_{i=1}^{n} \left(\frac{C_i}{T_i} + 1\right) \leq 2$$

High priority (2, 6)

Low priority (4, 8)

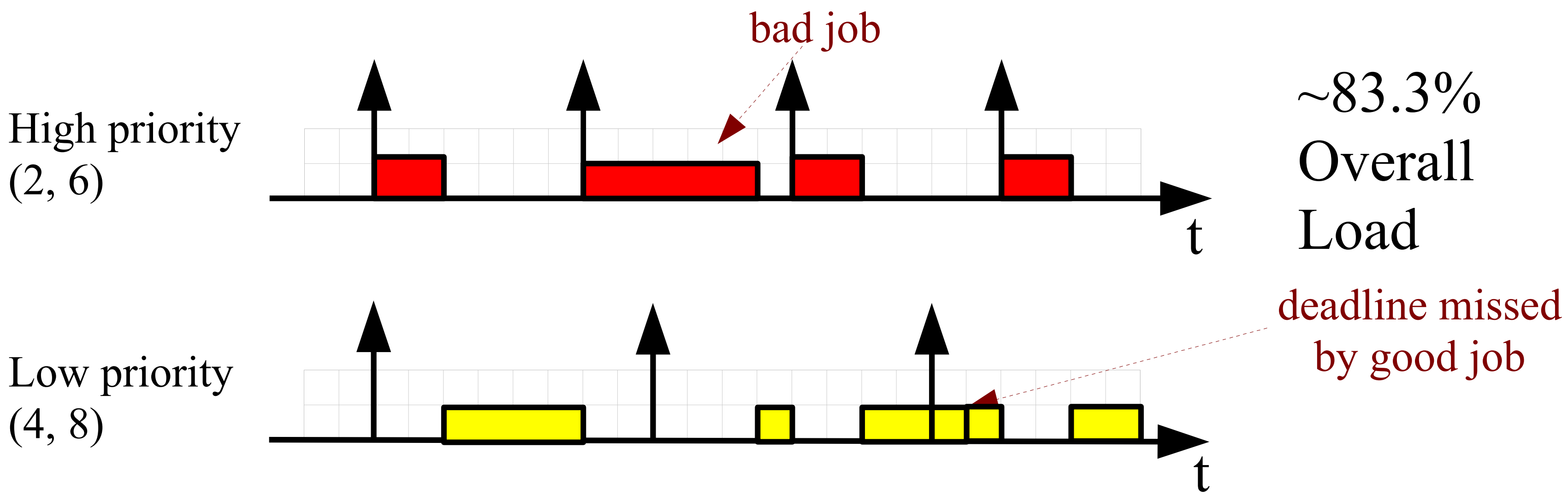


~83.3% Overall Load

Problems of Priority Scheduling

High-priority processes may indefinitely delay low-priority ones

- Coherent with the typical real-time/embedded scenario
  - Higher-priority processes are **more important** (e.g., safety critical)
- What if processes have same importance/criticality ?



Deadline-based Scheduling

Optimum for single-processor systems

- Necessary and sufficient admission control test for simple task model:

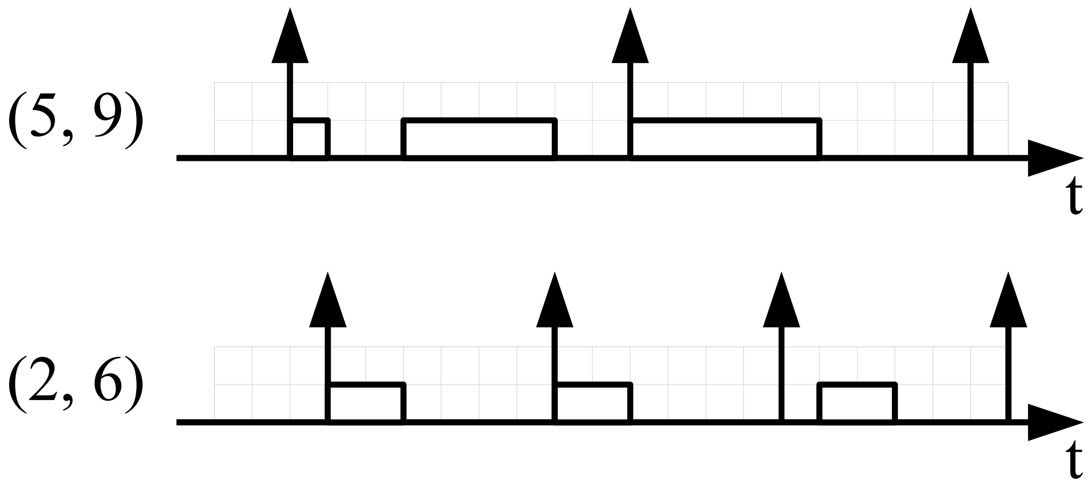
$$\sum_{i=1}^{n} \frac{C_i}{T_i} \le 1$$

Same problems of PS

- Deadlines respected as far as the WCETs are respected
- Things may go bad when
  - One or more tasks exhibit higher computation times than foreseen
  - One or more tasks behaves differently than foreseen
    - e.g., it blocks on a critical section for more than foreseen
- The task that suffers may not be the misbehaving one

Reservation-based scheduling: (Q_i, P_i)

- “Q_i time units **guaranteed** on a CPU every P_i time units”

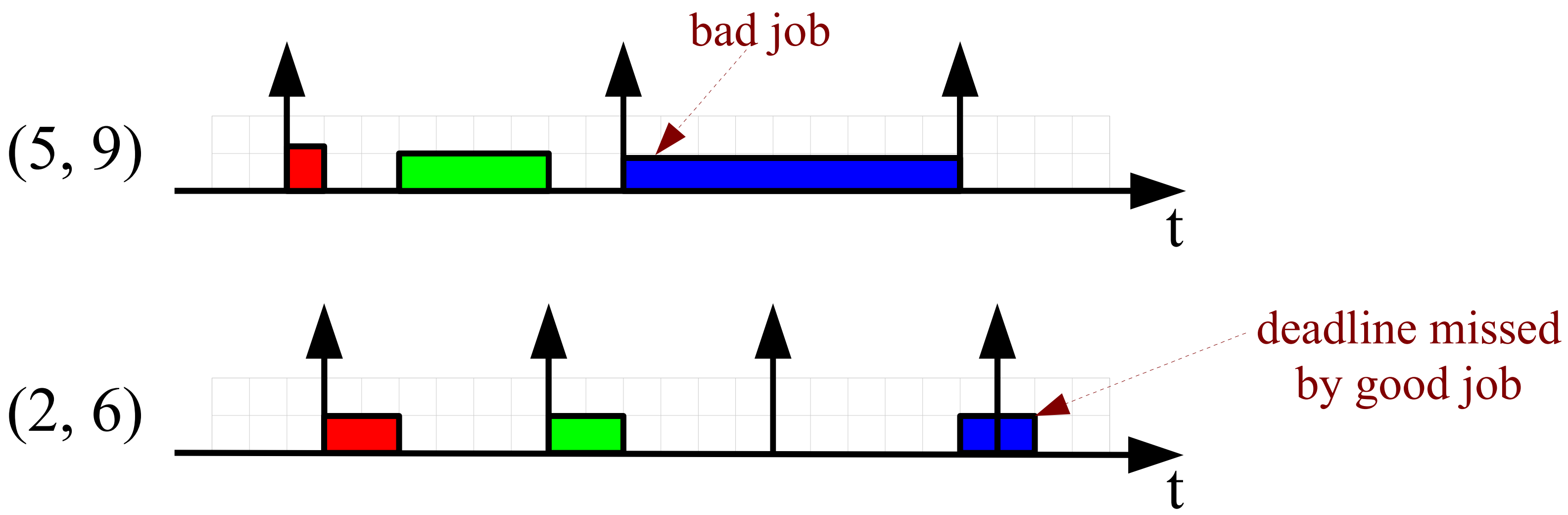


~88.9% Overall Load

- Independently of how others behave (**temporal isolation**)

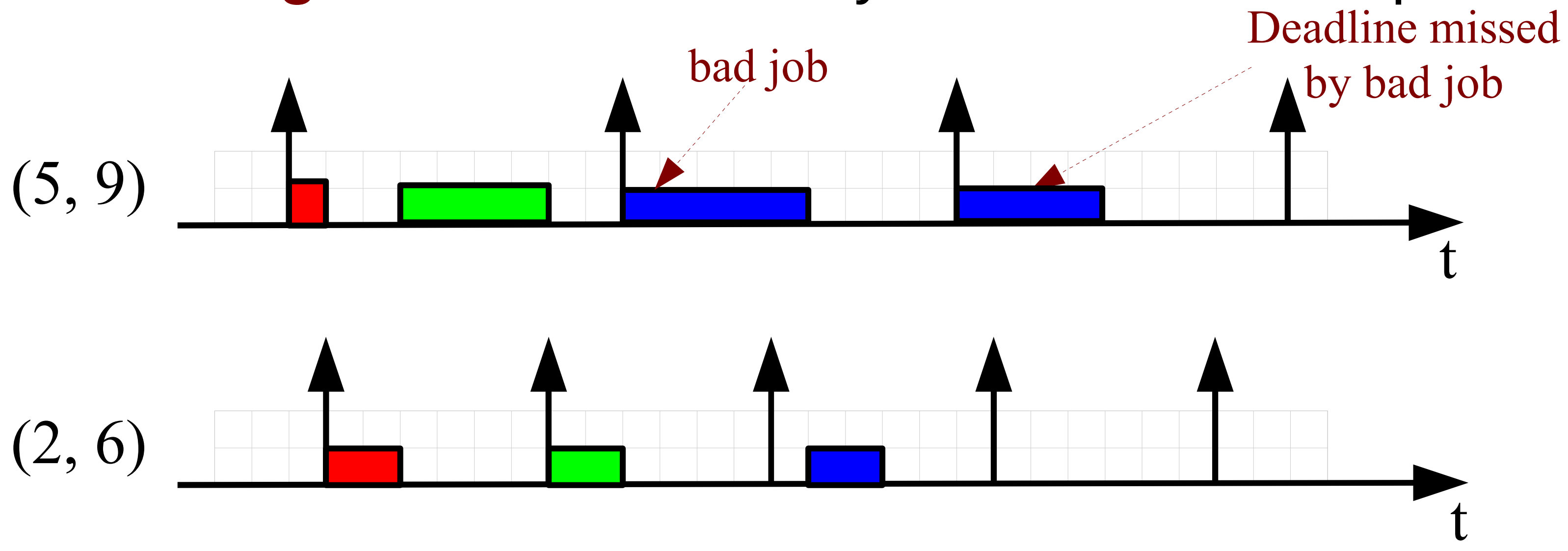
Enforcement of temporal isolation

- Not only EDF scheduling



Enforcement of temporal isolation

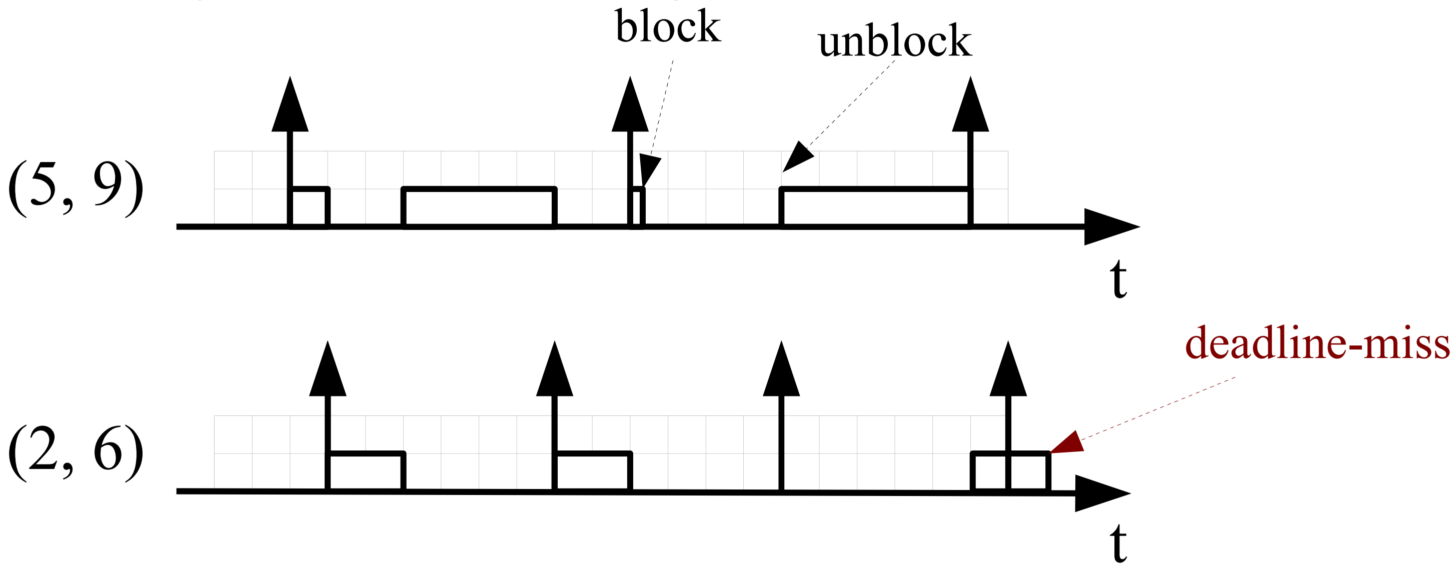
- Once **budget exhausted**, delay to next activation period



Temporal Isolation

Is needed despite blocks/unblocks

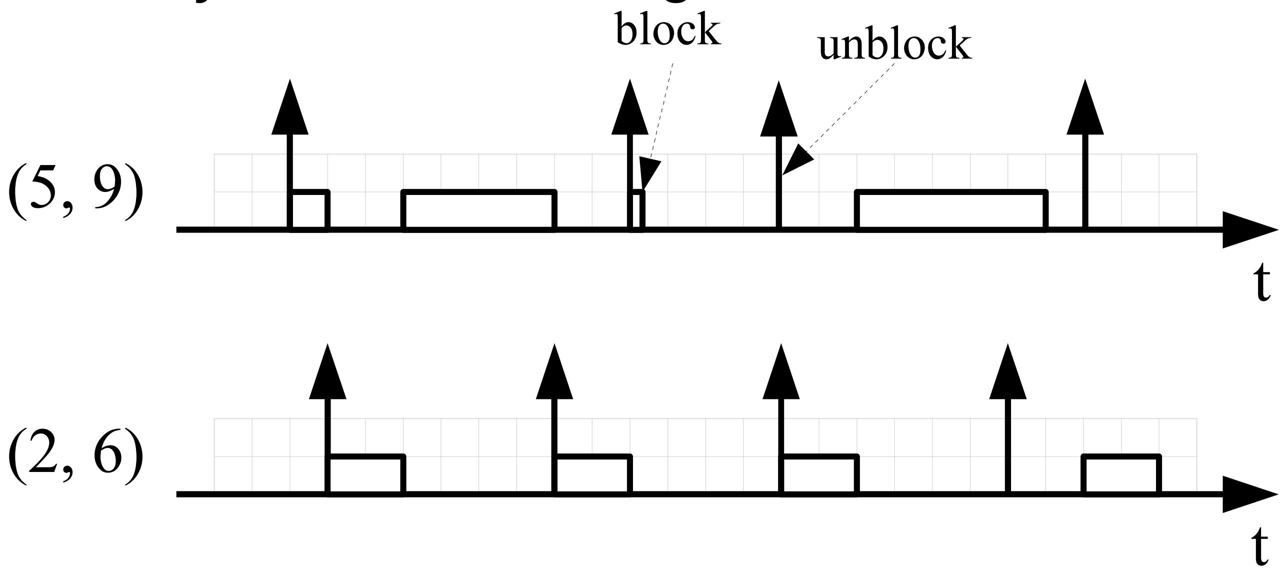
- Not only EDF scheduling



Temporal Isolation

Is needed despite blocks/unblocks

- Not only EDF scheduling

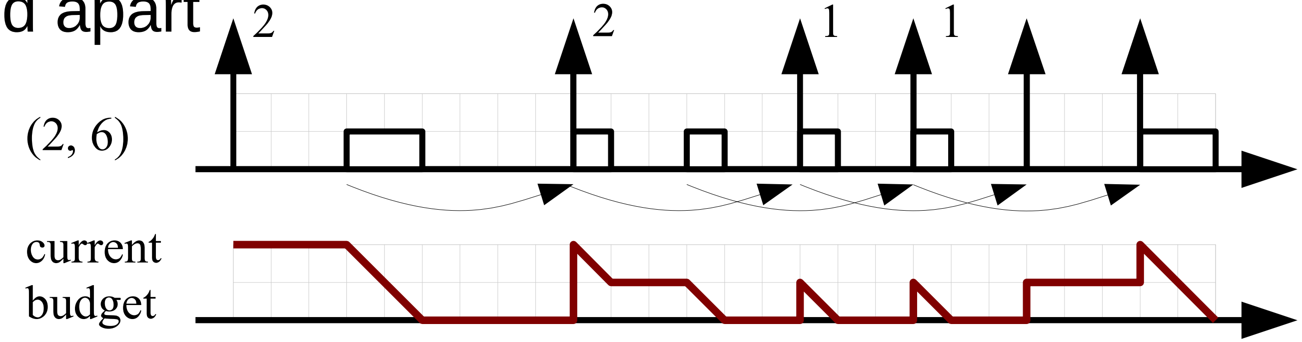


See CBS "unblock rule"

POSIX Sporadic Server

SCHED_SS

- Provides a form of temporal isolation
- Parameters: (Q, P, RT Priority, Low RT Priority)
- Budget exhausted => lower the priority till next recharge
- For every time interval in which the task executes, post a recharge of budget equal to the consumed CPU time one period apart



SCHED_SS may be analysed using FP techniques

- Patching the standard for getting rid of the “bug”

Process Scheduling in Linux

Real-Time Scheduling in Linux

Linux

- ➢ Not a Hard Real-Time OS
- ➢ Monolithic structure
  - Device drivers may adversely affect responsiveness

Advances in Linux temporal behavior (responsiv.)

- ➢ **Preemptability** of kernel-space code
- ➢ **High-resolution timers**
- ➢ Increasing use of **RCU** primitives
- ➢ Nearly complete support for **POSIX real-time** extensions
  - **Sporadic Server is missing (!)**
- ➢ **IRQ handlers as kernel threads** (preempt-rt branch)

Linux Scheduling Policies

POSIX compliant OS

- Implements priority-based scheduling
- Implements almost all real-time extensions
- Does not implement Sporadic Server
  - SSSA has an implementation available (still) as a separate patch
- pthreads compliant multi-threading support
  - The kernel deals with "**tasks**"

Goes beyond POSIX

- Support for multi-processor and multi-core systems (affinity)
- Support for NUMA machines
  - Scheduling domains with control over cpu sets and memory banks
- …

Linux Default Scheduler

Default Scheduler

- Design principles
  - **Round-Robin** policy as a starting base, for **fairness** among tasks
  - **Heuristic** to dynamically identify and **boost interactive (real-time)** tasks as compared to **batch** processing ones
  - Allow user-space to distinguish more/less important tasks
    - By setting their "**nice level**"
- Actual implementation changes from time to time
  - Completely priority-based, inefficient O(n) scheduler, 2.4 kernels
    - priority = nice level + dynamic priority offset (+/- 5)
  - Very Efficient O(1) scheduler, from 2.6.x
  - Efficient O(log n) Completely Fair Scheduler (CFS), from 2.6.y
    - Nice level corresponds to a weight in a (kind of) weighted RR

Linux and T.I.

Need for T.I. evident in mainline Linux

- A (buggy) real-time task can **starve** the entire OS
- **Real-time throttling** prevents this to happen

Real-Time Throttling

- Different design principles
  - Priority-based scheduling
  - Constraints to “no more than Q_i every system-wide P”
  - Behaves like “Deferrable Servers”

Thanks for your attention

Questions ?

Making Linux a better place

(for multimedia and soft real-time applications)

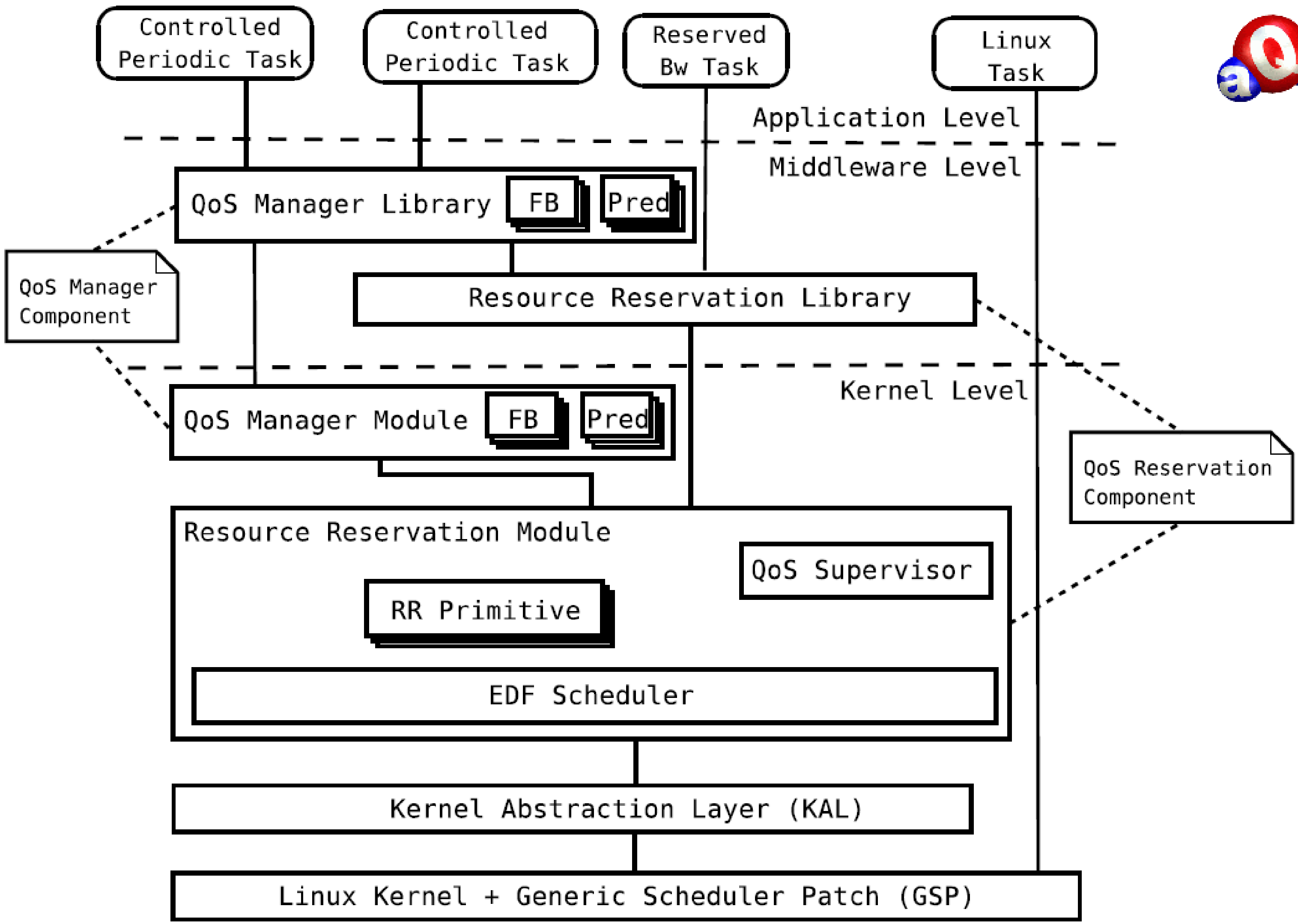
Research on Linux for Real-Time Systems at RETIS

- Applying techniques from Real-Time Systems to a GPOS
- **Improving responsiveness** and stability of performance for time-sensitive applications
- Investigating effectiveness of **Adaptive Reservations**
- **Synchronization** protocols for real-time multimedia
- Stabilising performance of **virtualized applications**
- **Isolating** computing and **I/O traffic** for Virtual Machines
- Effective exploitation of **multi-cores** in real-time systems
- **Programmability** of multi-threaded, distributed, real-time applications (API)

Resource reservation scheduling in Linux

- Adaptive Quality of Service Architecture
  - Single-processor embedded systems
  - **Multi-threaded** applications
- Hybrid Deadline/Priority Scheduler
  - **Multi-processor** systems
  - **Multi-threaded virtualized** applications
- Partitioned/Global EDF Scheduler SCHED_DEADLINE
  - **Multi-processor** systems
  - **Single-threaded control** applications

Adaptive Quality of Service Architecture for Linux



Framework for Real-time Embedded Systems based on COntRacts

Soft and hard real-time

Support for CPU, disk, network

Distributed real-time systems

Portability across RTOSs

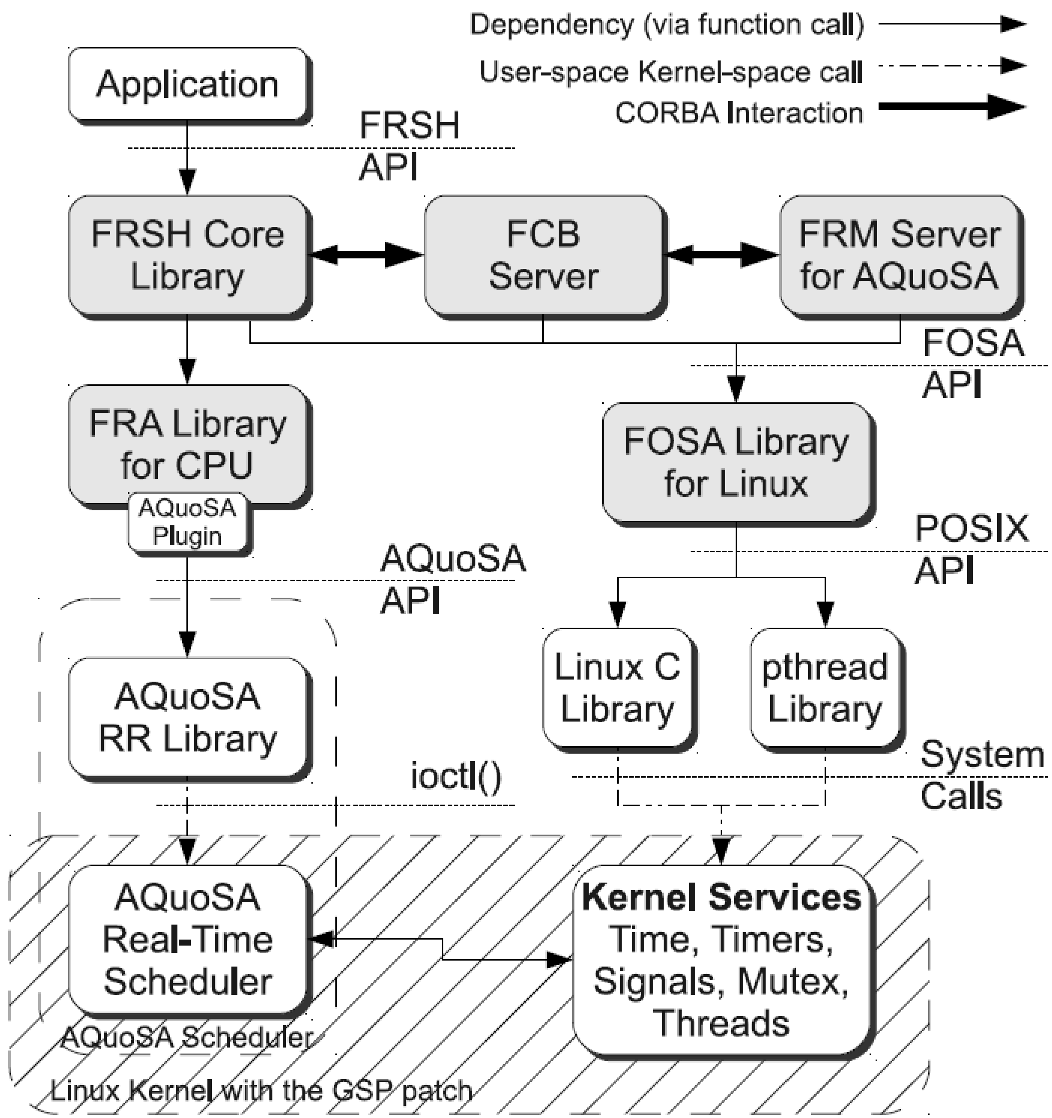
Adaptive real-time systems

Application-level QoS control

QoS power-aware optimization

Atomic negotiations

frescor



http://www.frescor.org

The IRMOS Scheduler

IRMOS

Interactive Realtime Multimedia Applications
on Service Oriented Infrastructures

IRMOS Scheduler

Features at a glance

- Resource Reservations
  - EDF-based scheduling (hard CBS)
- **Hierarchical scheduling**
  - Multiple tasks attached to same reservation
  - POSIX Fixed Priority scheduling inside each reservation
- **Multi-processor** reservations
  - Partitioned scheduling for improved efficiency
  - Migration of tasks among CPUs
- Simple **admission control**

IRMOS Real-Time Scheduler Design Goals

Replace real-time throttling

Tight integration in Linux kernel

- Modification to the Linux RT scheduler

Reuse as many Linux features as possible

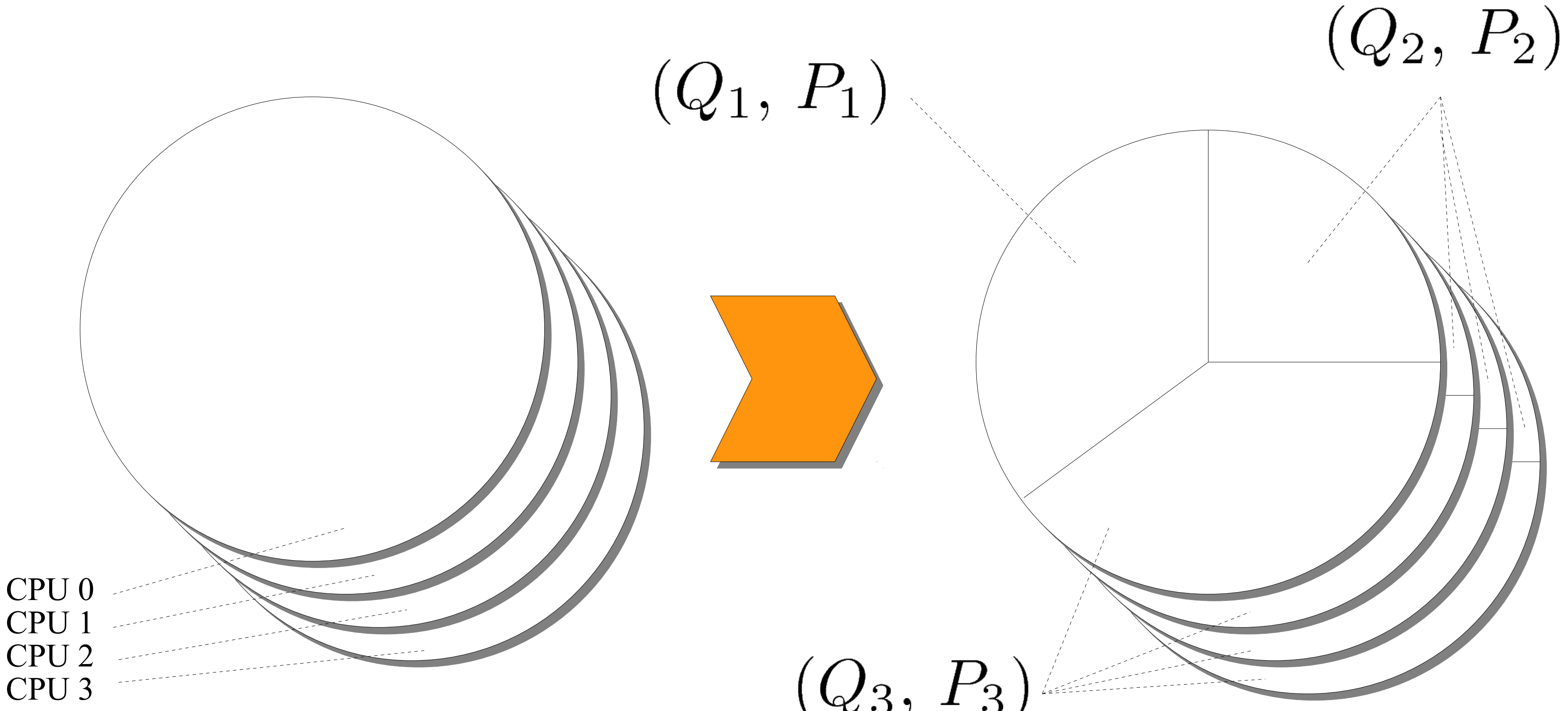
- Management of task hierarchies and scheduling parameters via **cgroups**
- POSIX compatibility and API

Efficient for SMP

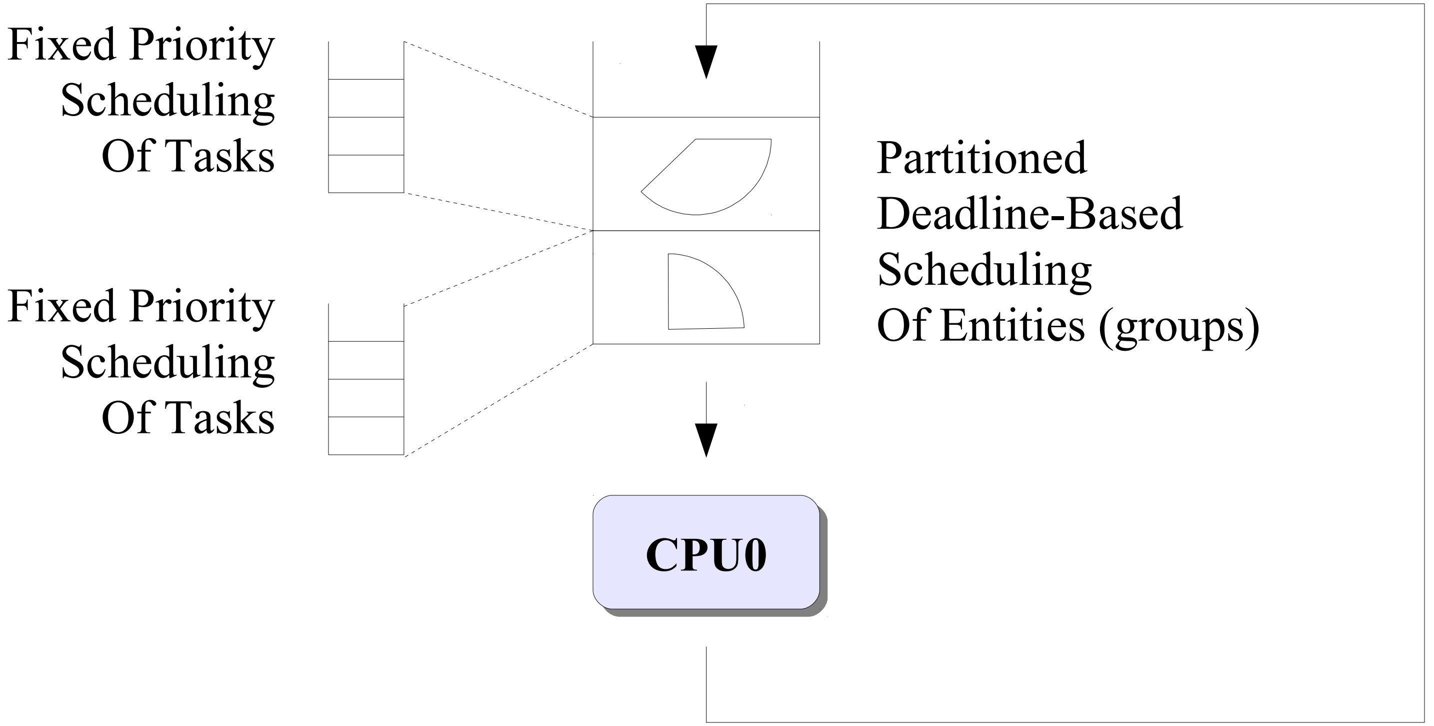
- Independent runqueues

IRMOS Scheduler

Slice the available computing power into reservations

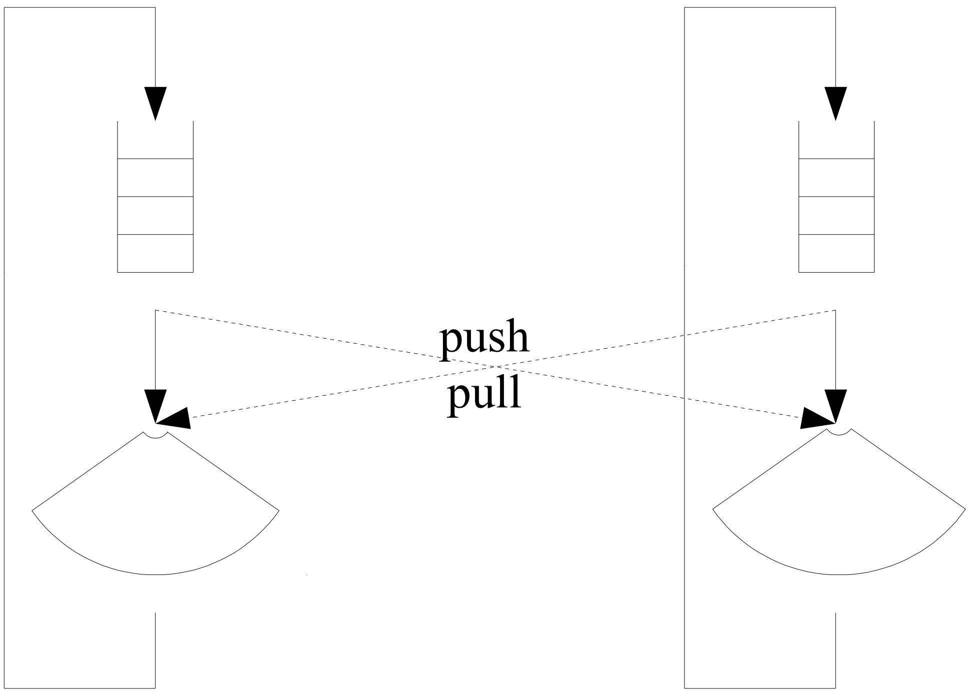


Partitioned CBS



Group-wide POSIX Fixed-Priority

- SCHED_RR, SCHED_FIFO both possible
- With M CPUs, if N ≤ M partitioned reservations are scheduled, then the N highest priority tasks in the group concurrently run



IRMOS Real-Time Scheduler

Short Demo

Applications Places System mer ott 6, 14:59 2,67 GHz

rt-app -P 4000 -d 4200

```
time=2996966, avg delay=940, max delay=1239 period=4000
time=3996971, avg delay=971, max delay=3443 period=4000
time=4996828, avg delay=965, max delay=1482 period=4000
time=5996987, avg delay=971, max delay=1243 period=4000
time=6996993, avg delay=982, max delay=1263 period=4000
time=7996828, avg delay=985, max delay=1223 period=4000
time=8997010, avg delay=997, max delay=1337 period=4000
```

```
tommaso@mobiletom:~$ man vncserver
tommaso@mobiletom:~$ man Xvnc
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$ run-xterm-rtapp.sh
tommaso@mobiletom:~$ run-xterm-rtapp.sh
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$
tommaso@mobiletom:~$ run-xterm-rtapp.sh
tommaso@mobiletom:~$
tommaso@mobiletom:~$ run-xterm-rtapp.sh
```

Owl Intranet Engine, Version Owl 0.96a 20081202

Trova: Precedente Successivo Evidenzia Maiuscole/minuscole

Download WOSS_2010_r... Pulisci

Completato zotero

tommaso... [Errore cari... Errore cari... Posta in ar... Composizi... Owl Virtes ... rt-app -P 4...

Applications Places System mer ott 6, 14:58 2,67 GHz

rt-app -P 4000 -d 4200

```
time=3996845, avg delay=934, max delay=4444 period=4000
time=4997014, avg delay=1057, max delay=8445 period=4000
time=5997015, avg delay=1003, max delay=4446 period=4000
time=6997012, avg delay=1006, max delay=4445 period=4000
^[[23~time=7997017, avg delay=994, max delay=1489 period=4000
time=8996863, avg delay=916, max delay=1824 period=4000
time=9996863, avg delay=927, max delay=3437 period=4000
```

```
A VNC server is already running as :0
tommaso@mobiletom:~$
tommaso@mobiletom:~$ man vncserver
```

rt-app -P 40000 -d 55000

```
time=3972131, avg delay=12151, max delay=12325 period=40000
time=4972151, avg delay=13040, max delay=25061 period=40000
time=5972130, avg delay=12171, max delay=12518 period=40000
time=6972136, avg delay=12159, max delay=12520 period=40000
time=7972137, avg delay=12208, max delay=12751 period=40000
time=8972145, avg delay=12176, max delay=12546 period=40000
time=9972136, avg delay=12186, max delay=12517 period=40000
```

```
tommaso@mobiletom:~$
tommaso@mobiletom:~$ run-xterm-rtapp.sh
```

Owl Intranet Engine, Version Owl 0.96a 20081202

Trova: Precedente Successivo Evidenzia Maiuscole/minuscole

Download WOSS_2010_r... Pulisci

Completato zotero

tommas... [Errore c... Errore c... Posta in ... Composi... Owl Virt... rt-app -P... rt-app -P...

rt-app -P 4000 -d 4200

```
time=1996772, avg delay=777, max delay=799 period=4000
time=2996772, avg delay=777, max delay=803 period=4000
time=3996772, avg delay=778, max delay=1172 period=4000
time=4996788, avg delay=817, max delay=929 period=4000
time=5996854, avg delay=788, max delay=941 period=4000
time=6996855, avg delay=872, max delay=1243 period=4000
time=7996790, avg delay=846, max delay=959 period=4000
```

```
[    10][qres_set_params_imp     ]<DBG>  Using Q=1200, P=4000, BW= 0.300000/0.750000
[    11][qres_set_params_imp     ]<DBG>  Creating new sever
```

rt-app -P 40000 -d 55000

```
time=1972095, avg delay=12097, max delay=12118 period=40000
time=2972095, avg delay=12102, max delay=12145 period=40000
time=3972096, avg delay=12099, max delay=12133 period=40000
time=4972269, avg delay=12224, max delay=12300 period=40000
time=5972269, avg delay=12271, max delay=12313 period=40000
time=6972268, avg delay=12260, max delay=12303 period=40000
time=7972267, avg delay=12259, max delay=12289 period=40000
```

```
irmos/bin/rt-app -P 4000 -d 4200
[    15][qres_cleanup            ]<DBG>  Cleaning up
```

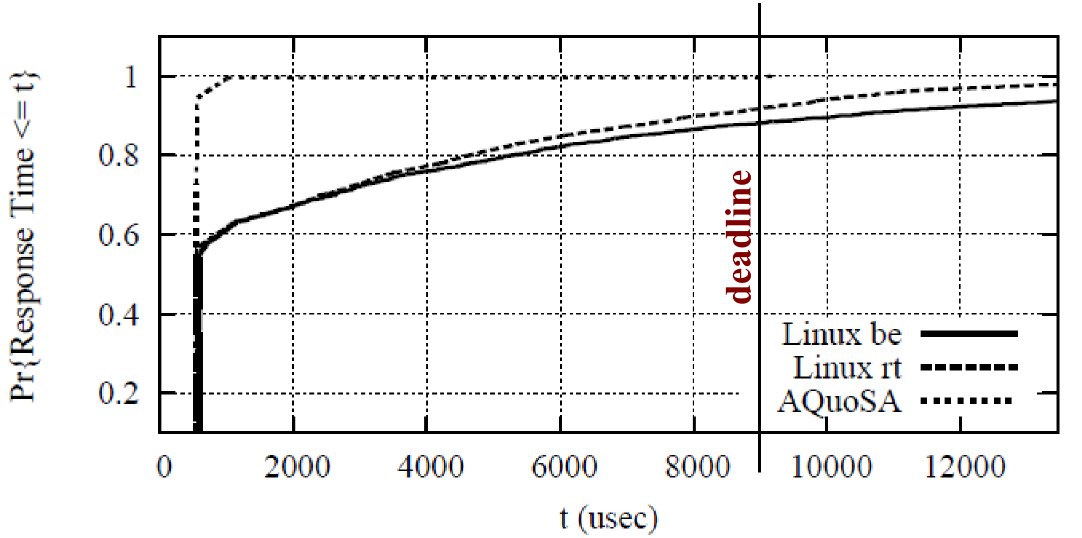
Owl Intranet Engine, Version Owl 0.96a 20081202

Let's Give Some Numbers

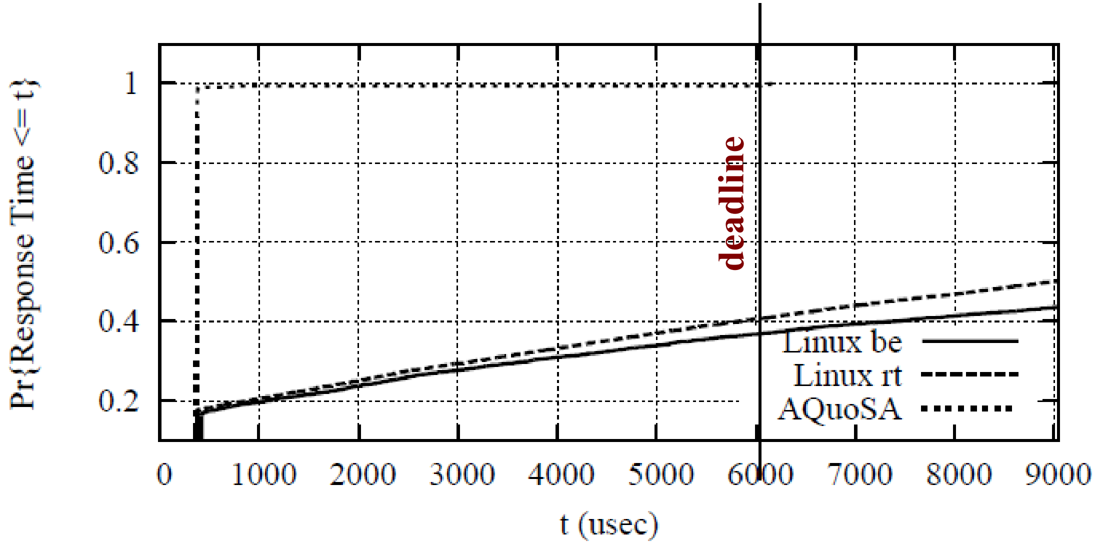
Experimental results

(Pentium 4 @ 2GHz, Linux 2.6.29.1 + AQuoSA)

Response-time CDF for the real-time task with the shortest period under light load (48%)



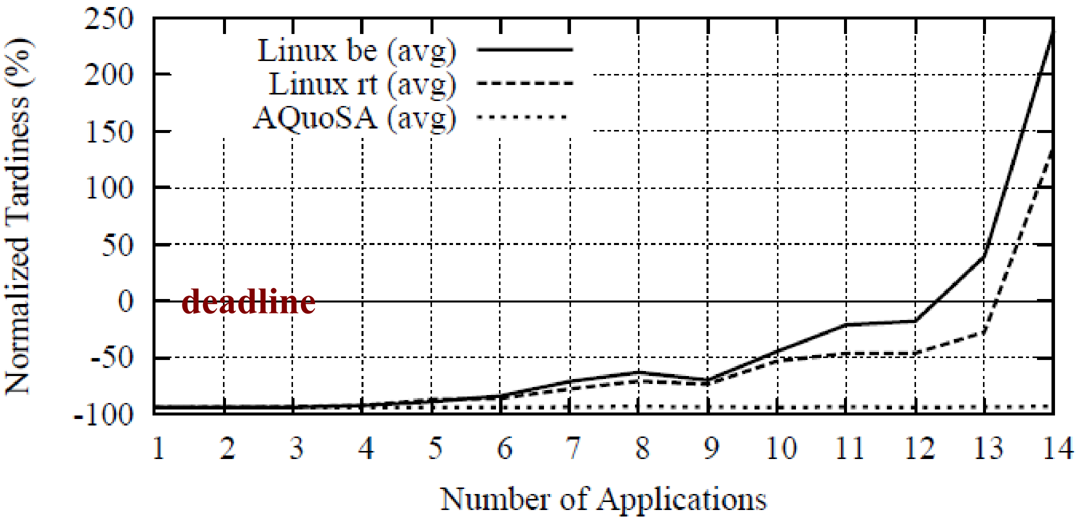
Response-time CDF for the real-time task with the shortest period under heavy load (84%)



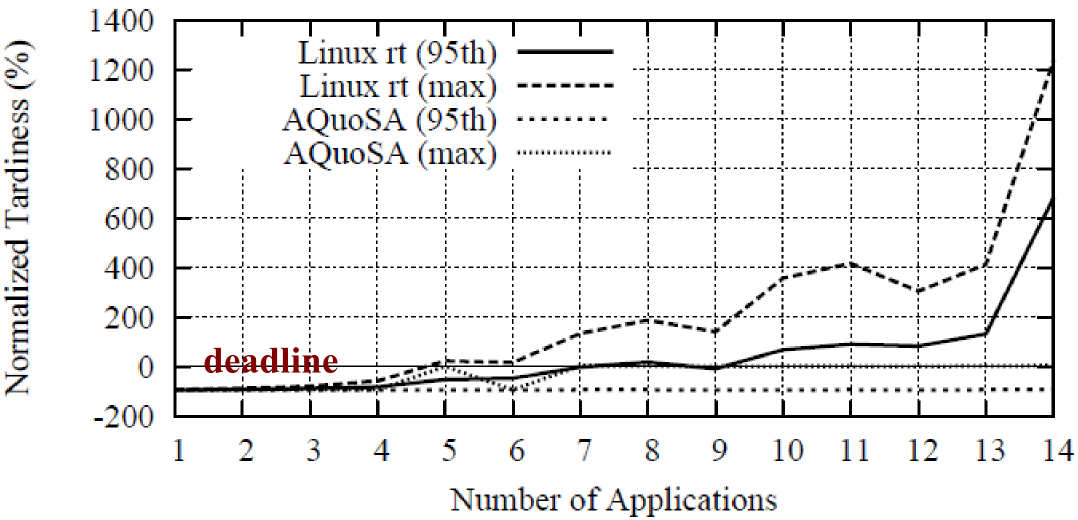
Experimental results

(Pentium 4 @ 2GHz, Linux 2.6.29.1 + AQuoSA)

Average normalized tardiness for the real-time task with the shortest period, varying the number of real-time tasks in the system.



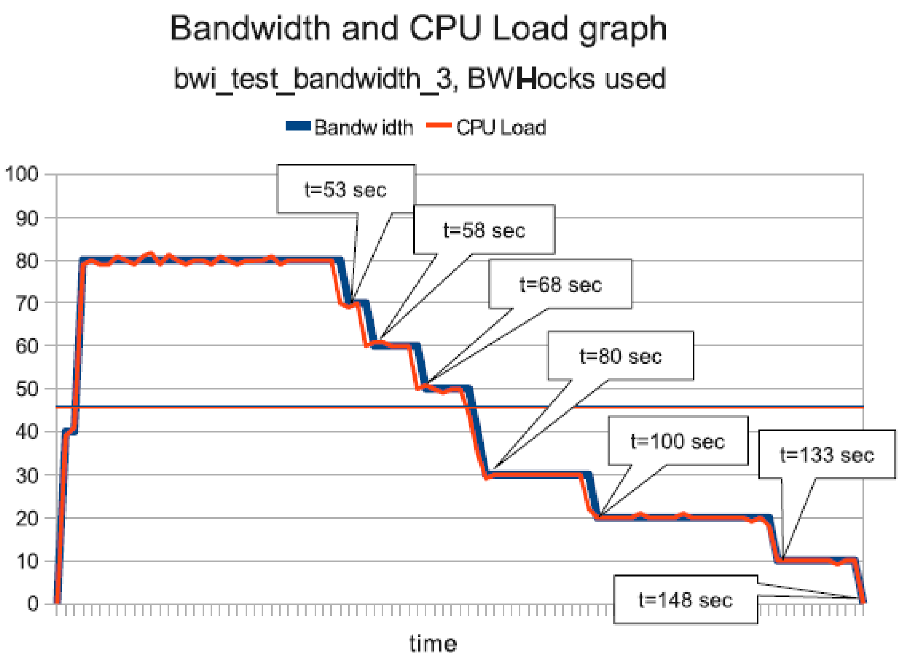
Maximum and 95th percentile of the normalized tardiness for the real-time task with the shortest period, varying the number of real-time tasks in the system.

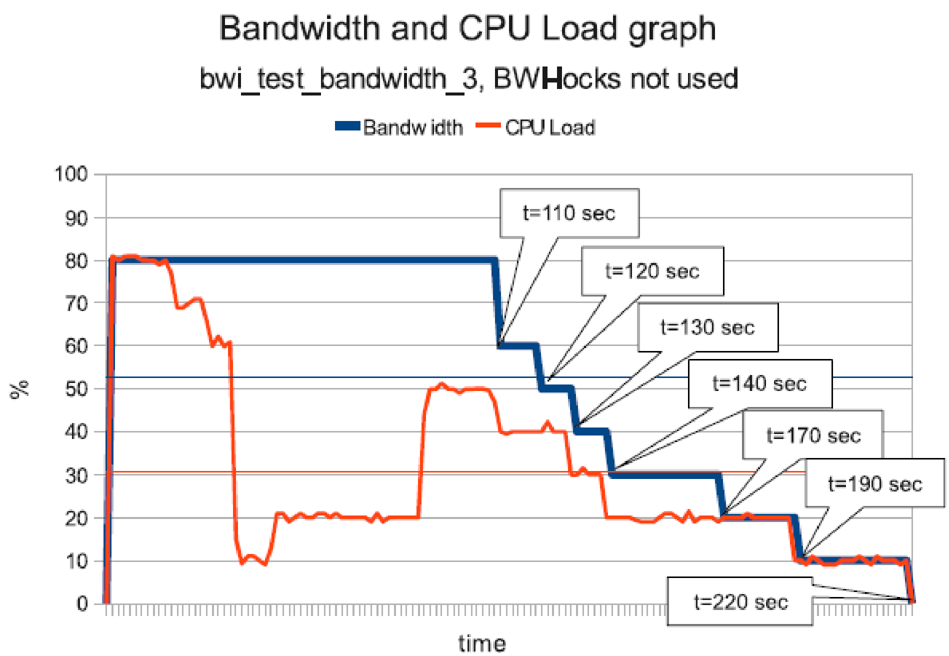


Synchronization

Bandwidth Inheritance (BWI / M-BWI)

- Deals with **Priority Inversion** in EDF scheduling
- **Deadline inherited** from lock owner





Thanks for your attention

Questions ?

Virtualized Real-Time Applications

(Real-Time IaaS)

Introduction

Virtualization & real-time increasingly interesting

- Wide availability of broadband connections ==> shift in computing paradigms towards distributed computing (**cloud computing**)
  - Not only *remote storage* and *batch processing*
  - But also *remote processing* for *interactive applications*
- Examples
  - **Virtual Reality** with heavyweight physics simulations
  - Distributed editing of HD video (**film post-production**)

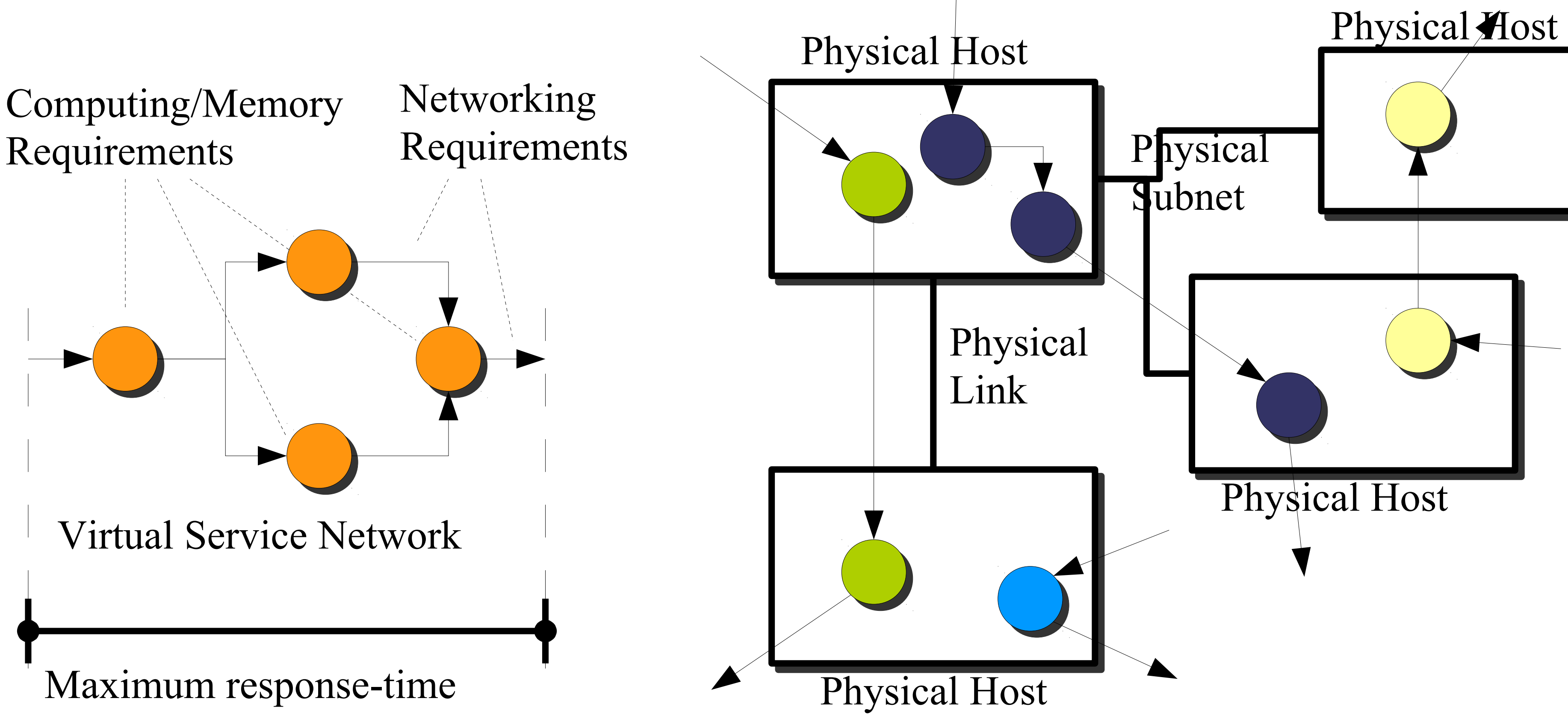
Service-Oriented Architectures

- Promising approach to distributed computing
- Taking advantage of virtualization techniques:
  - Location independence
  - Security
  - Fault-Tolerance
- Distributed Interactive/Real-Time Applications may benefit from SOA design

Problem presentation

Optimum/reasonable deployment of VSNs on PNs

- Given computing/network/memory requirements
- Respecting end-to-end timing constraints



Problem presentation

Issues in deploying RT SW Components in VMs

- Scheduling and timing
  - **VM scheduling impacts on the vision of time by guest OSes**
    - Time granularity (for measuring time and setting timers)
    - Non-uniform progress-rate of applications
  - SMP-enabled guests
    - Spin-lock primitives assume release of locks within very short time-frames
      - What happens if the **lock-owner VM is descheduled** ?
- **Benchmarking**
  - A VM may be deployed on different HW (SOA scenario)
    - How to achieve predictable performance ?
  - VMs may be deployed on **General-Purpose HW** (with cache)
    - How to account for **HW-level interferences** ?

Problem presentation

Issues in deploying RT SW Components in VMs

- **Temporal isolation** across VMs
  - Compute-bound and I/O-bound VMs
  - Shared host resources (e.g., network interrupt drivers)
  - Intensive I/O on virtualised peripherals
- Proper management of shared resources:
  what MP resource-sharing protocol is appropriate ?
  - Proper management of **priority inversion**
  - Reduced overheads (limited number of preemptions)
  - Run-time schedulability analysis and **admission control**

Approach

Traditional (hard) real-time techniques are not appropriate

- lead to poor resource utilization
- imply high/unsustainable development costs

Soft real-time techniques are more appropriate

- **Stochastic models** for system/QoS evolution
- **Probabilistic guarantees** (as opposed to deterministic ones)

Pragmatic approach

- Theory is always applied
  - on **real GPOS** (Linux)
  - with a **real Virtual Machine Monitor** (KVM)
  - on **real multimedia applications**

Approach

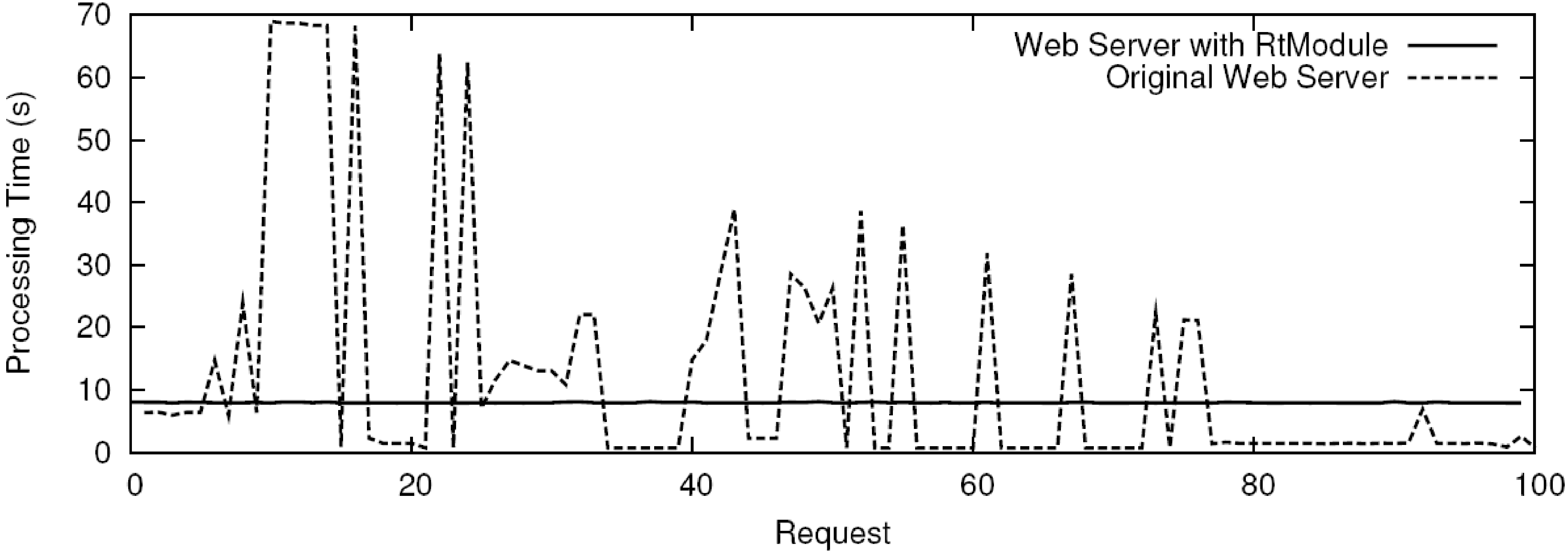
Basic Building blocks

- Linux Kernel as host OS enriched with our RT Scheduler(s)
- Each VMU is attached RT scheduling parameters (defining its temporal capsule)
- Improvements on the real-time virtualization performance
  - Modifications at the hypervisor level
  - Modifications at the kernel level
- Analysis of Virtualized Real-Time applications by Hierarchical Real-Time Schedulability Analysis

Experimental results (RTSOAA 2009)

Temporal isolation of compute-intensive VMUs

- Response-times of Apache2 Web Server
  - → From **unpredictable and highly variable**
  - → To **predictable and very stable** (low fluctuations)
    - – At the cost of an *increased minimum response-time*



Experimental results (VHPC 2010)

Temporal isolation of networking traffic among concurrent VMUs

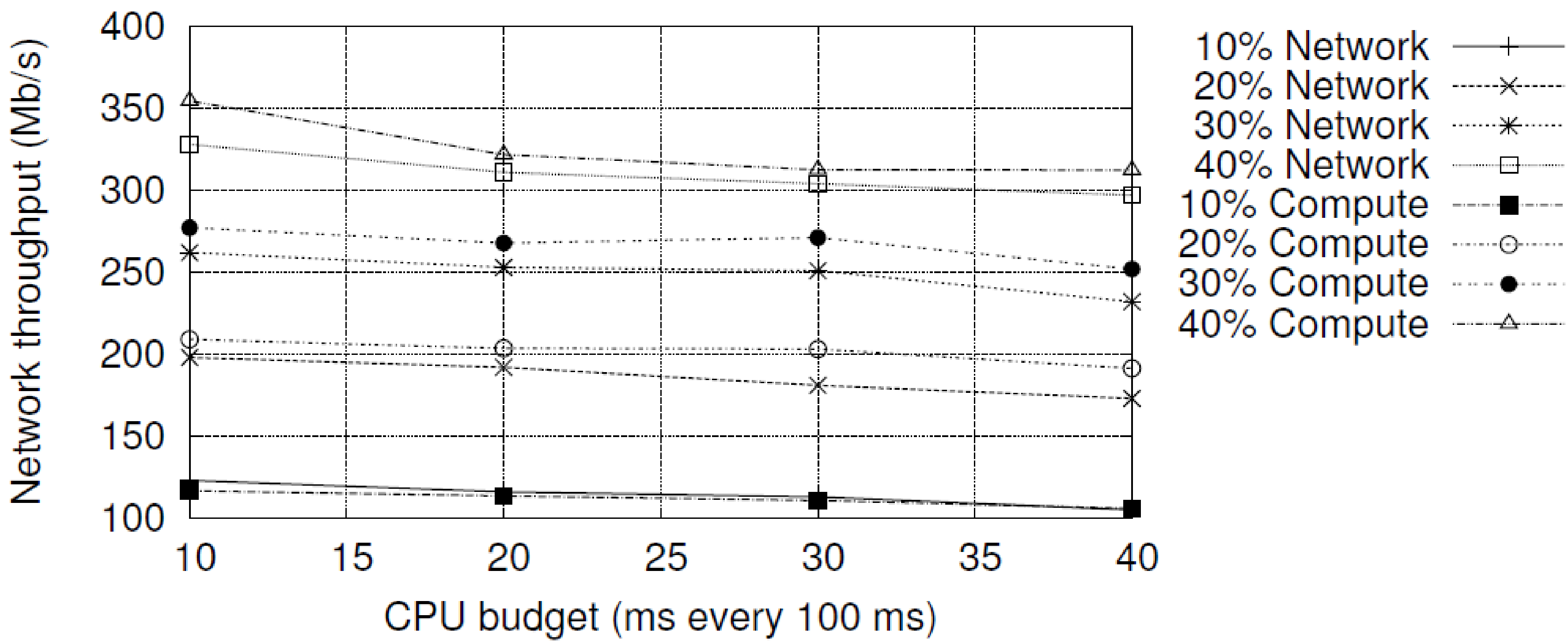


Fig. 3: Network throughput (Y axis) for a VM as a function of the CPU share of the other VM (X axis), at varying CPU shares for itself (different curves), in case of CPU- and I/O-intensive loads.

Theoretically Optimum Deployment

of

Distributed Real-Time Workflows

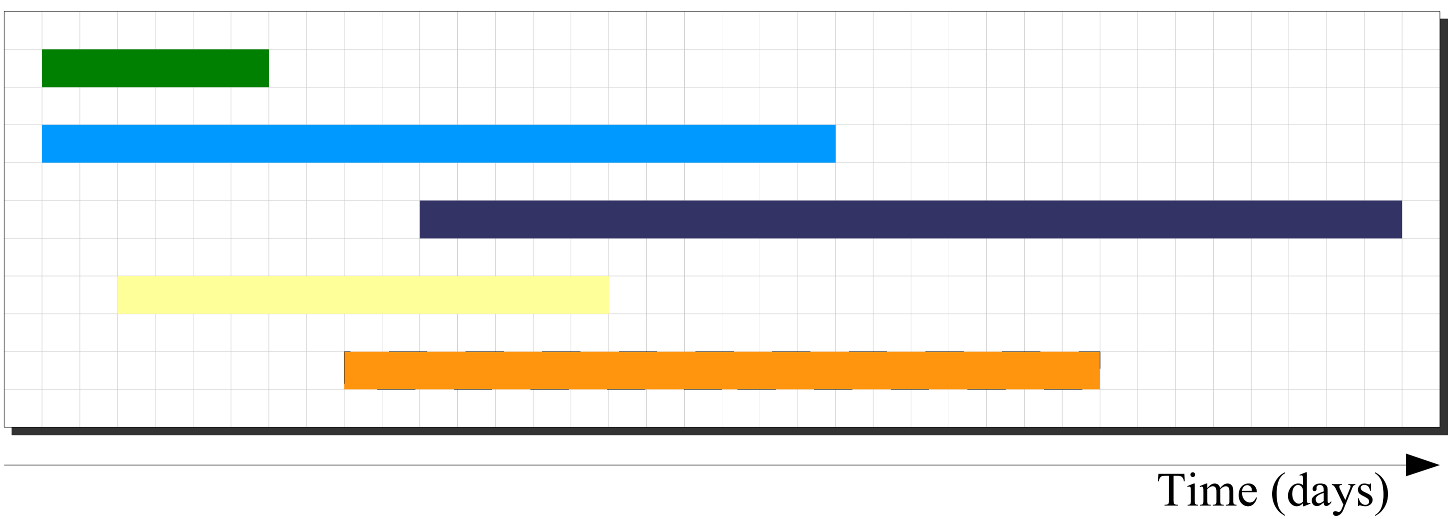
with

End-to-end Response Requirements

Problem presentation

Optimum deployment of VSNs on PNs

- Considering expected usage time-horizon (**advance reservations**)
- Periods of overlapping reservations



Proposed approach

Temporal isolation among independent application workflows

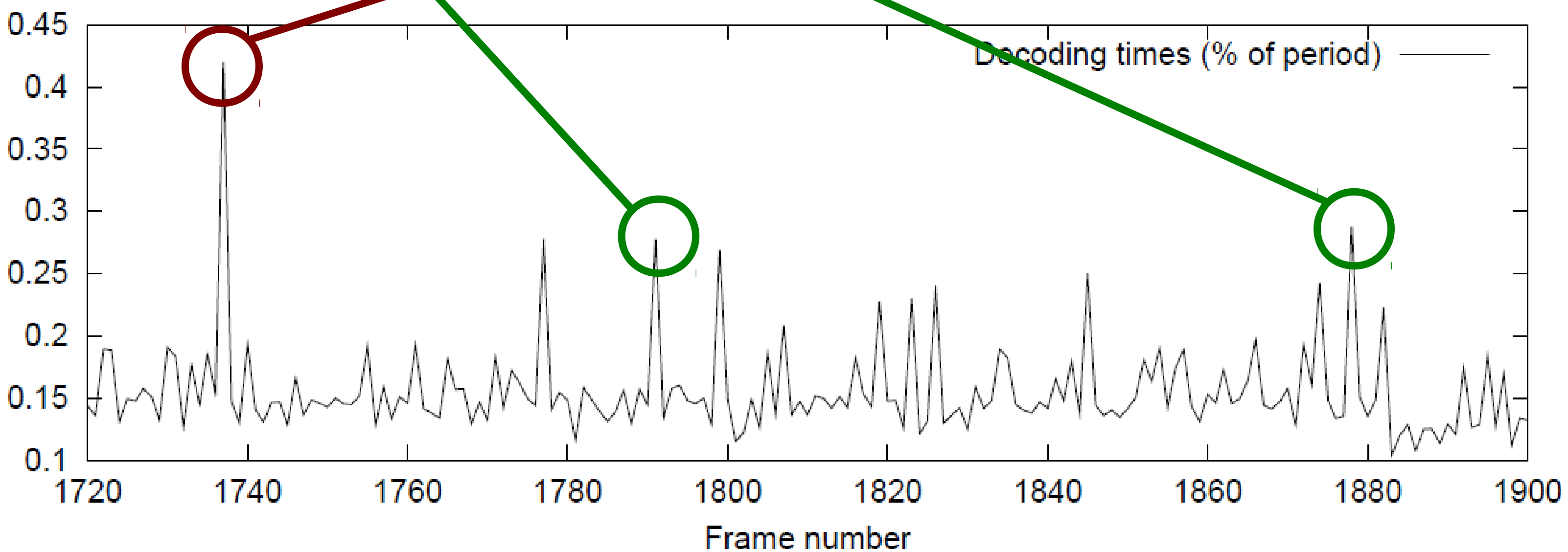
- Time-sharing of **heterogeneous** computing nodes
  - Through **real-time scheduling** at the OS/kernel level
- Time-sharing of network links
  - Through **QoS-aware scheduling** of the medium (e.g., Wf^2Q+)

How to tune resource allocation ?

- i.e., real-time scheduling parameters

Probabilistic availability guarantees

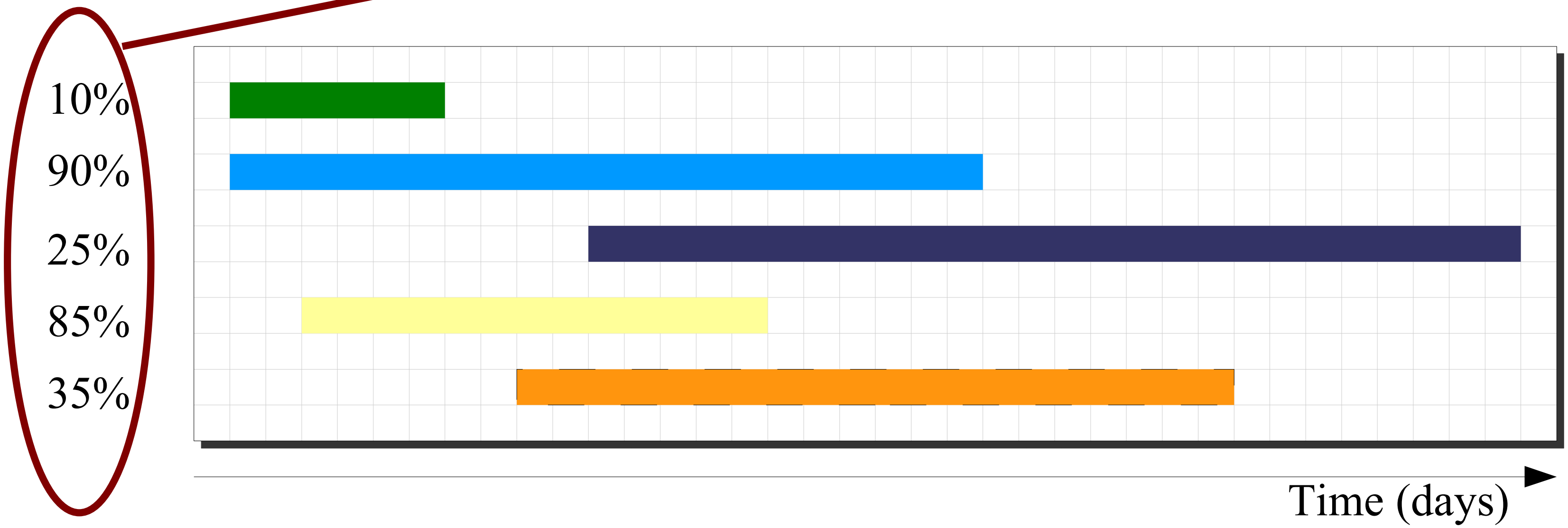
Tune allocation on computation-time percentile (instead of WCET)



Probabilistic availability guarantees

Applications sharing the same PH may be independently activated

Provider relies on actual probabilities of activation for admitted & new services



Modelling

Finally, we obtain a Mixed-Integer Non-Linear Programming (MINLP) optimization problem

- Constraints
  - Physical resources topology
  - Application (VSN) topology
  - Deterministic formulation
    - Maximum end-to-end latency for deployed workflows
    - Maximum saturation level for each physical host and link
  - Probabilistic formulation
    - Minimum probability of having enough resources when the application is actually activated by the user
    - Minimum probability of respecting the end-to-end latency
- Objective function
  - e.g., minimize number of used hosts, maximize provider's revenue

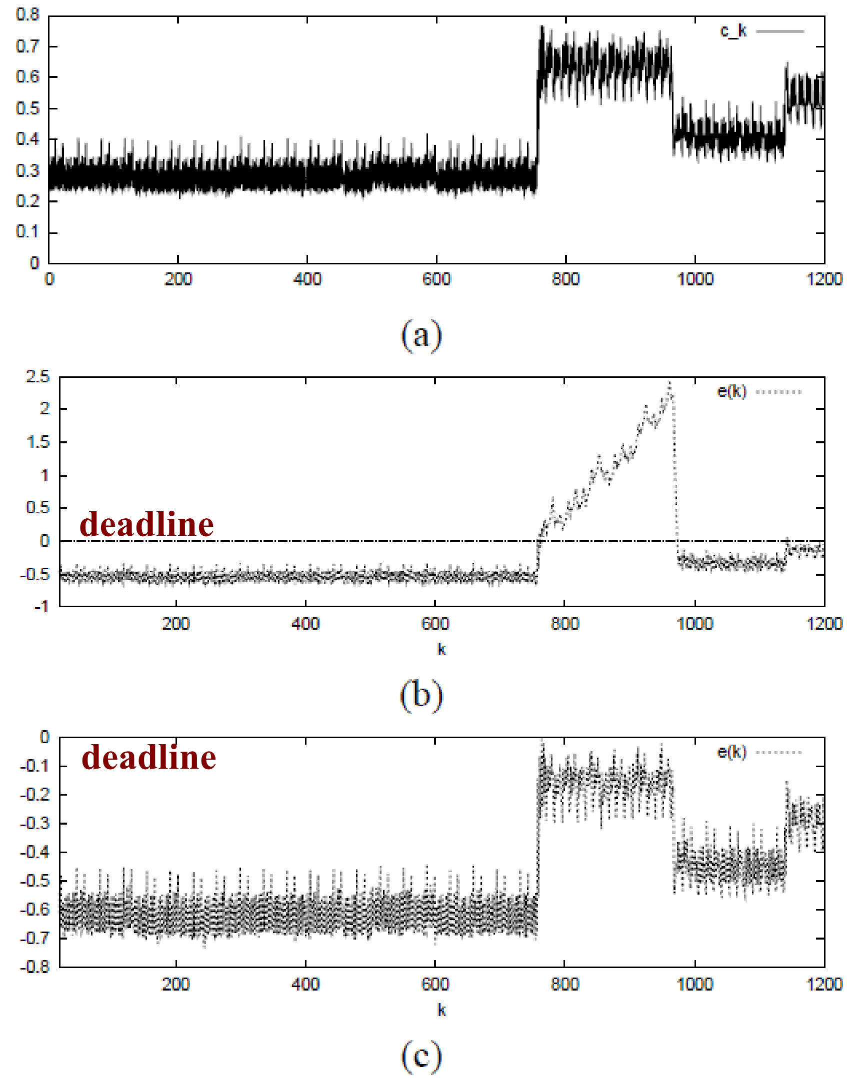
Adaptive Reservations

Need for adaptivity

Computation times for decoding MPEG frames

Scheduling error with an over-allocation of 30% over the average

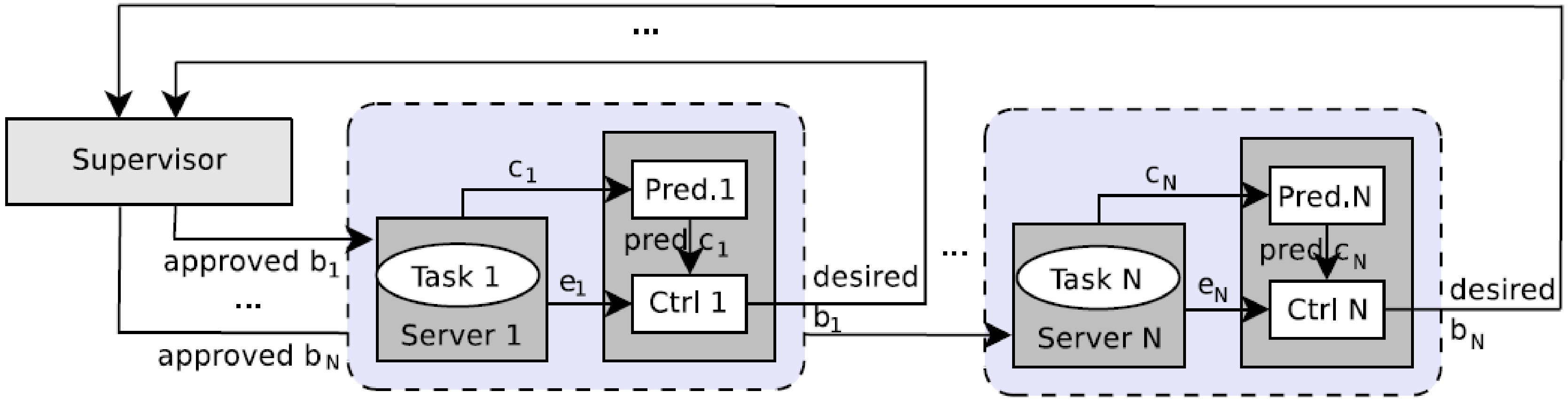
Scheduling error with allocation tuned on maximum



Adaptive Reservations

Feedback-based scheduling

- **Sense**: tracking of workload fluctuations
- **Compute**:
  - Prediction/estimation of workload for next period(s)
  - Compensation for possible current delays
- **Actuate**: adapt the scheduling parameters (Q_i, P_i)



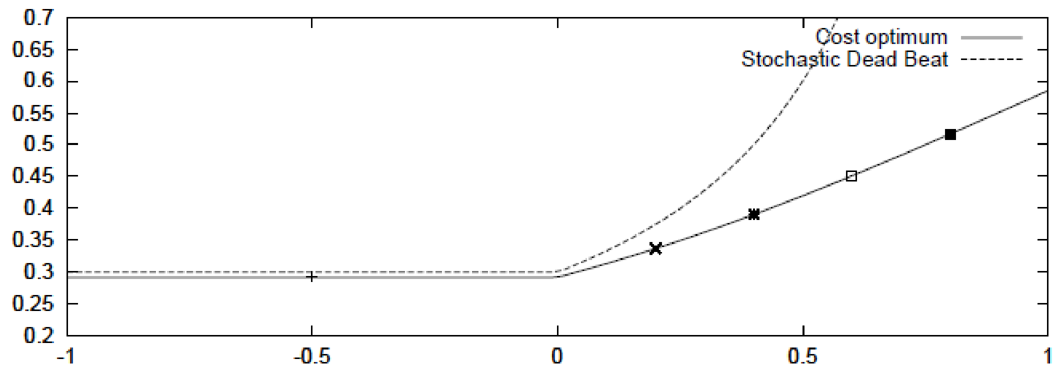
Adaptive Reservations

Workload prediction

- Moving average
- Percentile estimation over moving window
- FIR decorrelation + error estimation

Budget controller

- Target region around deadline
- Stochastic approaches
  - Probability of deadline-miss
  - Optimum error/bandwidth trade-offs



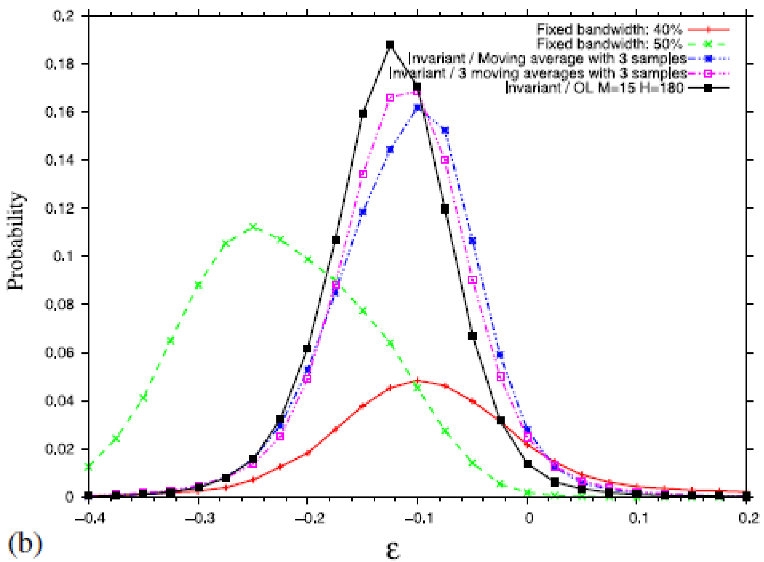
$$B_k(\varepsilon_k) = \sqrt[3]{\rho + \delta(\varepsilon_k)} + \sqrt[3]{\rho - \delta(\varepsilon_k)}$$

$$\rho = \frac{\gamma(\sigma^2 + \mu^2)}{(1-\gamma)}$$

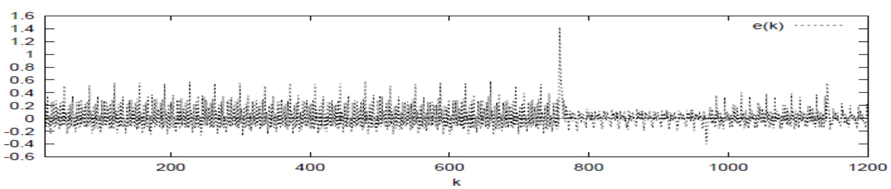
$$\delta(\varepsilon) = \sqrt{\left(\frac{\gamma}{1-\gamma}\right)^2 (\sigma^2 + \mu^2)^2 + \left(\frac{2}{3}\frac{\mu\gamma[1 - S(\varepsilon_k)]}{1-\gamma}\right)^3}$$

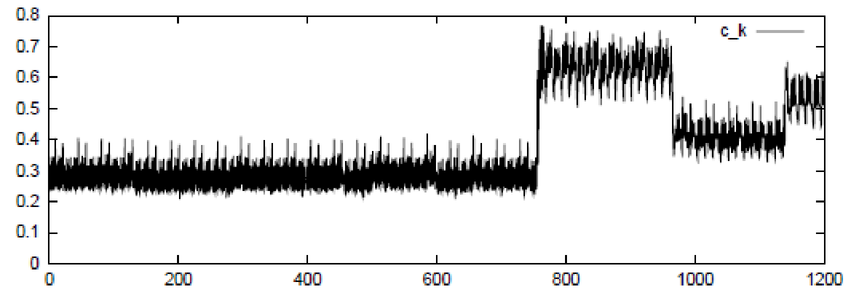
Experimental results

PMF of scheduling error with various budget control strategies

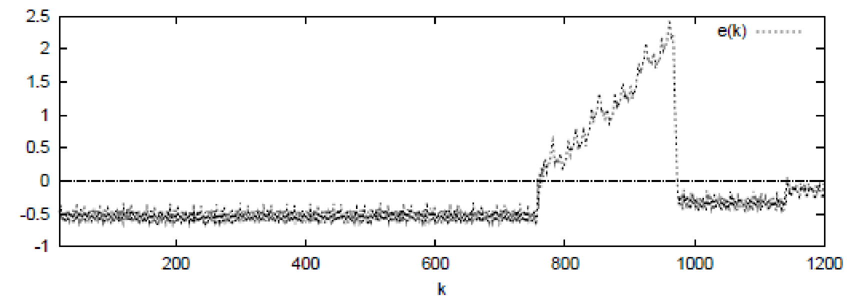


(b)

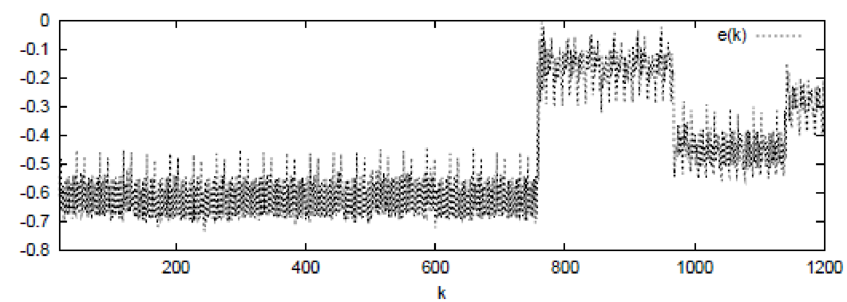




(a)



(b)



(c)

Automatic Identification of Scheduling Parameters

Automatic identification of scheduling parameters

Recent developments in GPOS CPU scheduling

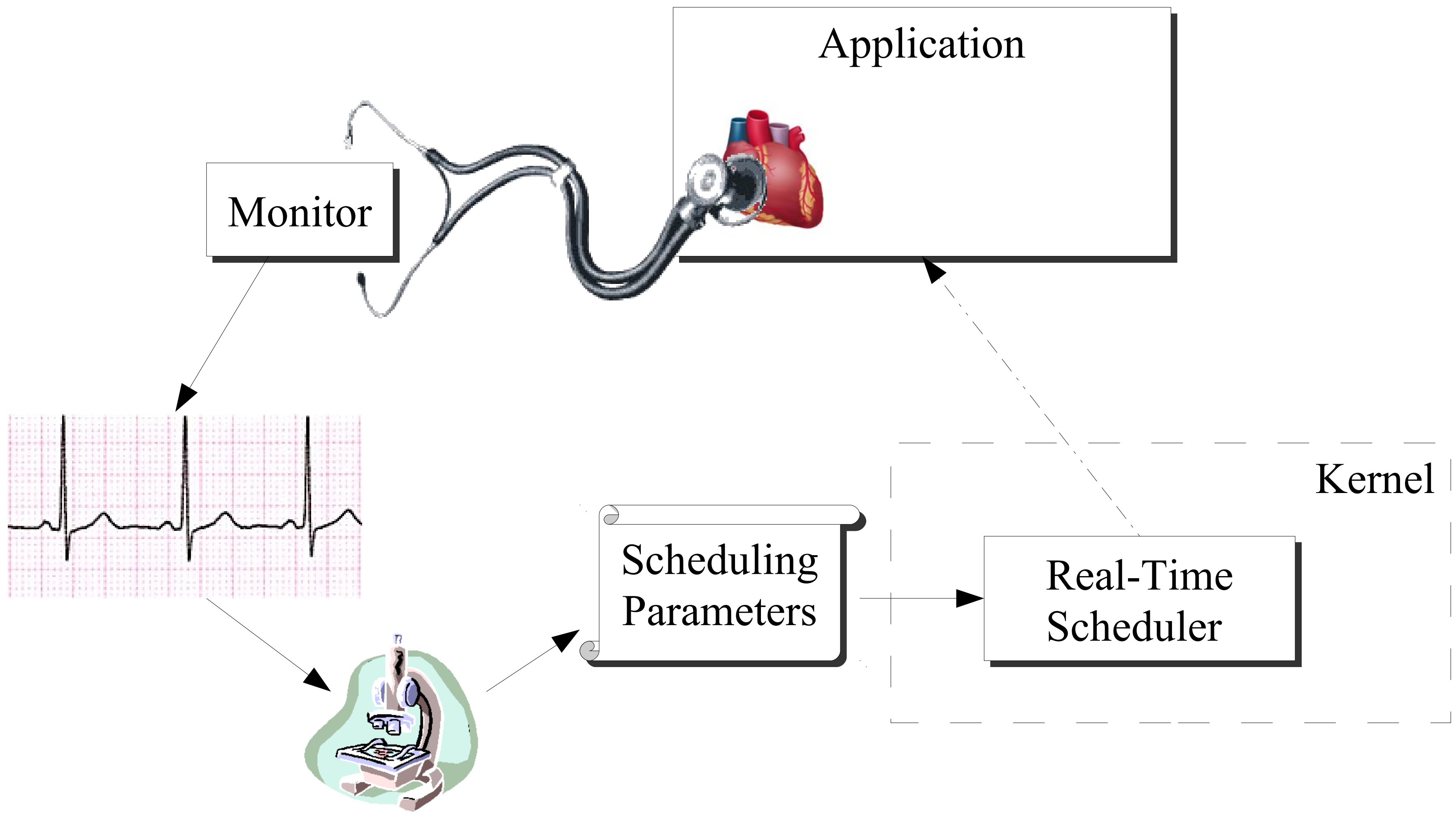
- Various APIs for accessing the enhanced functionality
  - For example, the FRSH API
  - For example, the AquoSA API
- They require **modifications** of the applications
  - at the **source-code level**
- Can we provide real-time guarantees to **unmodified** applications ?
  - For example, for **legacy multimedia** applications ?
  - Or, simply because of no time to modify the applications

Objectives

We actually can

- allow (legacy) real-time periodic applications
- to benefit of real-time scheduling facilities increasingly available on a GPOS
- **without any change** in the application source-code

LFS++


Application
Monitor
Kernel
Scheduling
Parameters
Real-Time
Scheduler

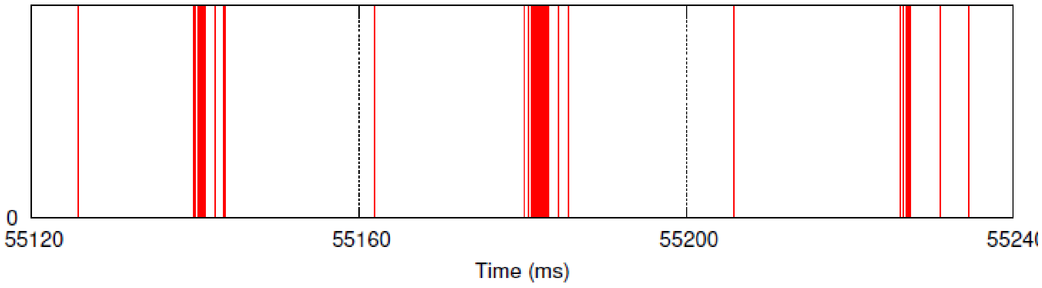
Proposed approach

Legacy Feedback Scheduling (LFS++)

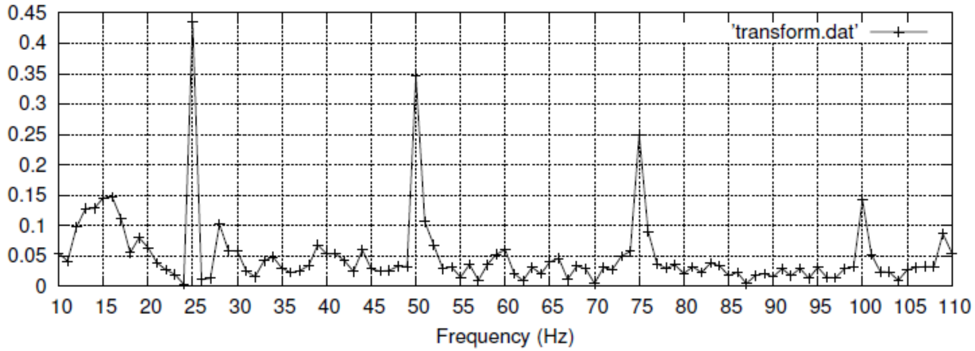
- An appropriate **tracing mechanism** observes the application, *inferring main parameters* affecting a (periodic) multimedia application temporal behaviour:
  - *job execution time*
  - **period**
- Scheduling guarantees are **automatically** provisioned by the OS, according to proper **scheduling parameters**
  - Based on sound arguments from real-time theory

Period detection

The tracer produces a sequence of time-stamps



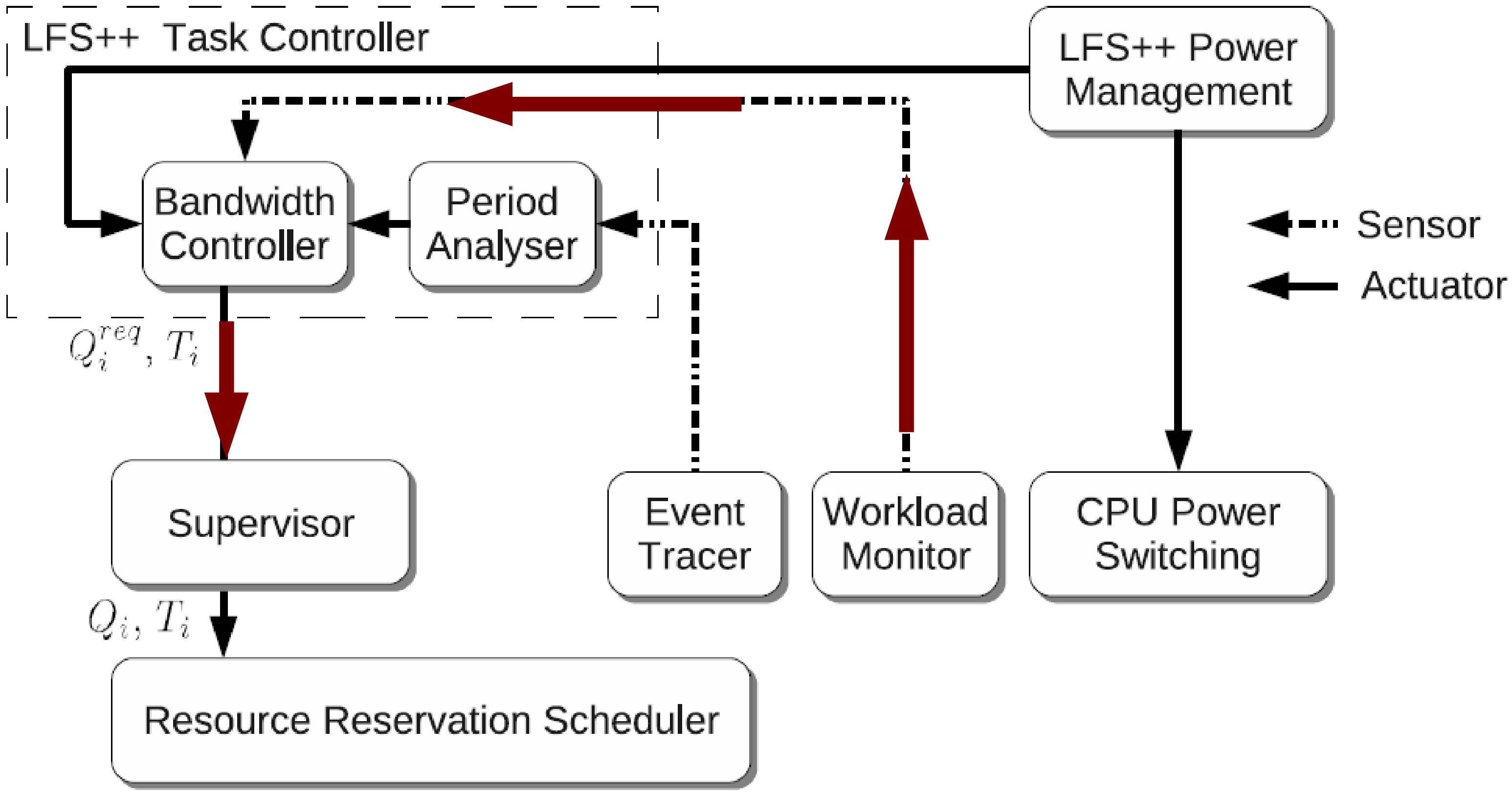
Time-stamps used to compute a Fourier-transform



A heuristic catches the first harmonic

Budget identification

"Feedback-based scheduling" budget control loop



Experimental results

Set-up

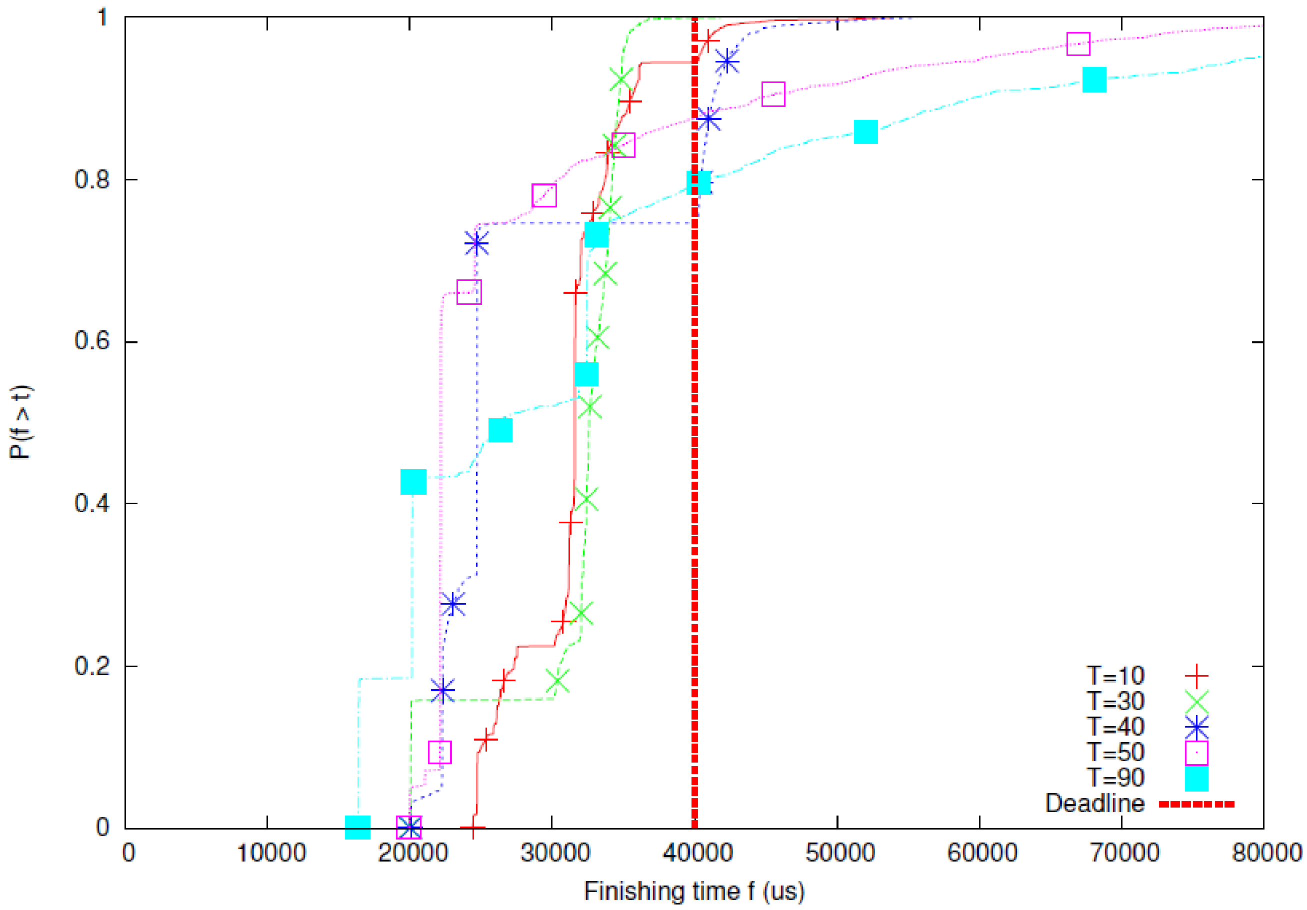
- Linux 2.6.29, with an implementation of the CBS scheduler
- Feedback-scheduling by means of AQuoSA
- **mplayer**
  - modified to monitor the **Inter-Frame Time (IFT)** and
- Application tracing by using **qtrace** (kernel-level)

Validation metrics

- **Inter-Frame Time** (for mplayer)
- **A/V desynchronisation** (for mplayer)
- **Response-time** (for synthetic application)
- **Allocated bandwidth** on the Real-Time scheduler

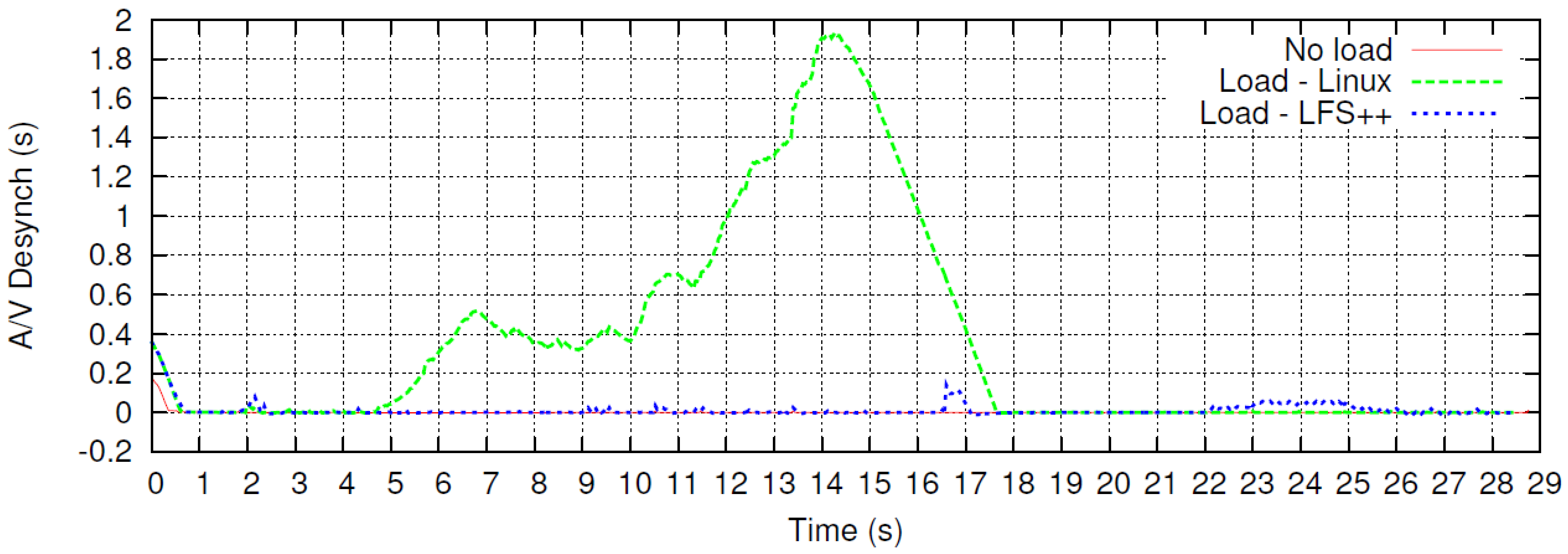
Experimental results

Using the correct reservation period is better



A/V desynchronisation in mplayer while starting the Eclipse IDE

- **90% reduction** of the peak A/V desynchronisation



Thanks for your attention

Questions ?

New Research Themes

at RETIS

Future and Emerging computing platforms

- **Massively parallel** processors
  - → Hundreds/thousands cores per-processor
  - → Hundreds/thousands processors per-system
- **Heterogeneity** in
  - → **Computing power** and capabilities
  - → Communication **latencies**
- Nowadays **Operating Systems inadequate** for
  - → **Massively parallel applications**
  - → With **temporal constraints** (e.g., performance / interactivity)
  - → Running on **massively parallel systems** (**no scalability**)

S(o)OS
Service-oriented Operating Systems

Issues to be investigated

- Partitioning of cores among functionality
  - → For reducing contention and enhancing cache efficiency
  - → e.g., **kernel cores**, **application cores**, **interrupt cores**, …
- Kernel based on **message-passing**, not **shared memory**
  - → For reducing contention in accessing shared kernel-level data
  - → A kernel instance per core
- Application-level control of (and API for):
  - → sharing level for kernel data (e.g., file descriptors)
  - → page table entries
- **Scalable synchronisation** primitives
- **Distributed** protocols for **in-chip resources allocation**
  - → e.g., spatial **scheduling**

Recent Publications (2009/2010)

Journals

- *A robust mechanism for adaptive scheduling of multimedia applications*, T. Cucinotta, L. Abeni, L. Palopoli, G. Lipari, to appear on ACM Transactions on Embedded Computing Systems
- *QoS Control for Pipelines of Tasks using Multiple Resources*, T. Cucinotta, L. Palopoli, IEEE Transactions on Computers, Vol. 53, No. 3, pp. 416--430, March 2010
- *A Real-time Service-Oriented Architecture for Industrial Automation*, T. Cucinotta, A. Mancina, G. F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, F. Rusinà, IEEE Transactions on Industrial Informatics, Vol. 5, n. 3, August 2009

Conferences/Workshops

- *Providing Performance Guarantees to Virtual Machines using Real-Time Scheduling,* T. Cucinotta, D. Giani, F. Checconi, D. Faggioli, VHPC 2010, August 2010
- *An Exception Based Approach to Timing Constraints Violations in RT and Multimedia Applications,* T. Cucinotta, D. Faggioli, IEEE SIES 2010
- *The Multiprocessor BandWidth Inheritance Protocol*, D. Faggioli, G. Lipari, T. Cucinotta, ECRTS 2010, July 2010
- *Advance Reservations for Distributed Real-TimeWorkflows with Probabilistic Service Guarantees*, T. Cucinotta, K. Konstanteli, T. Varvarigou, in SOCA 2009, December 2009, Taipei, Taiwan
- *Self-tuning Schedulers for Legacy Real-Time Applications*, T. Cucinotta, F. Checconi, L. Abeni, L. Palopoli, EuroSys 2010, Paris, April 2010
- *Hierarchical Multiprocessor CPU Reservations for the Linux Kernel*, F. Checconi, T. Cucinotta, D. Faggioli, G. Lipari, OSPERT 2009, Dublin, Ireland, June 2009
- *Respecting temporal constraints in virtualised services*, T. Cucinotta, G. Anastasi, L. Abeni, RTSOAA 2009, Seattle, Washington, July 2009

Thanks for your attention

Final questions ?

http://retis.sssup.it/people/tommaso