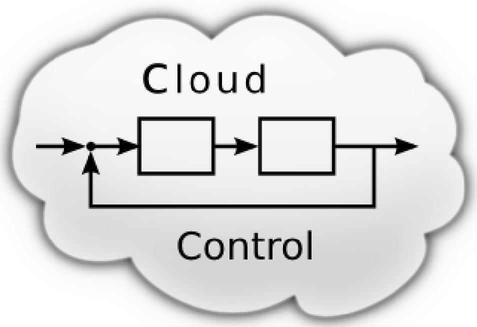
Dynamic Resource Management for Multicore Linux Platforms

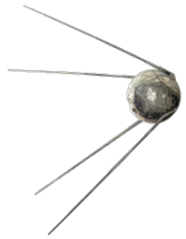
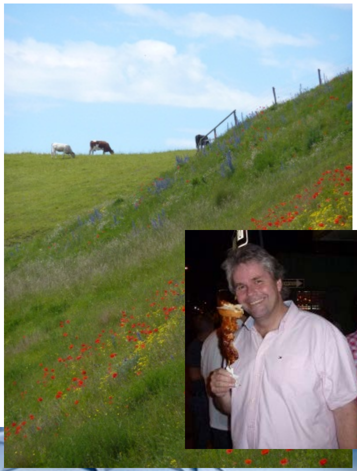
Karl-Erik Årzén
Lund University
Lund, Sweden

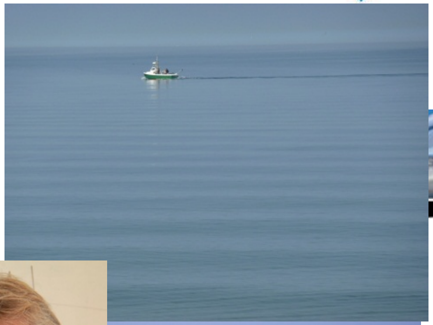


EMSIG Autumn PhD School 2015, Lyngby, Denmark

Karl-Erik Årzén

- **Private life:**
  - Born 4 Oct 1957 in Malmö
- **Professional life:**
  - E-77, Lund University, Sweden
  - PhD 87 – AI for Control
  - ABB Corporate Research
  - Professor in Automatic Control 2000
  - Co-PI of the Linneaus Center LCCC
  - Vice director of the ELLIIT SRA
  - Member of the Program Management Group of WASP
  - Chair of IVA Syd
- **Research Interests**
  - Embedded Control
  - Feedback Computing
  - Cloud Control

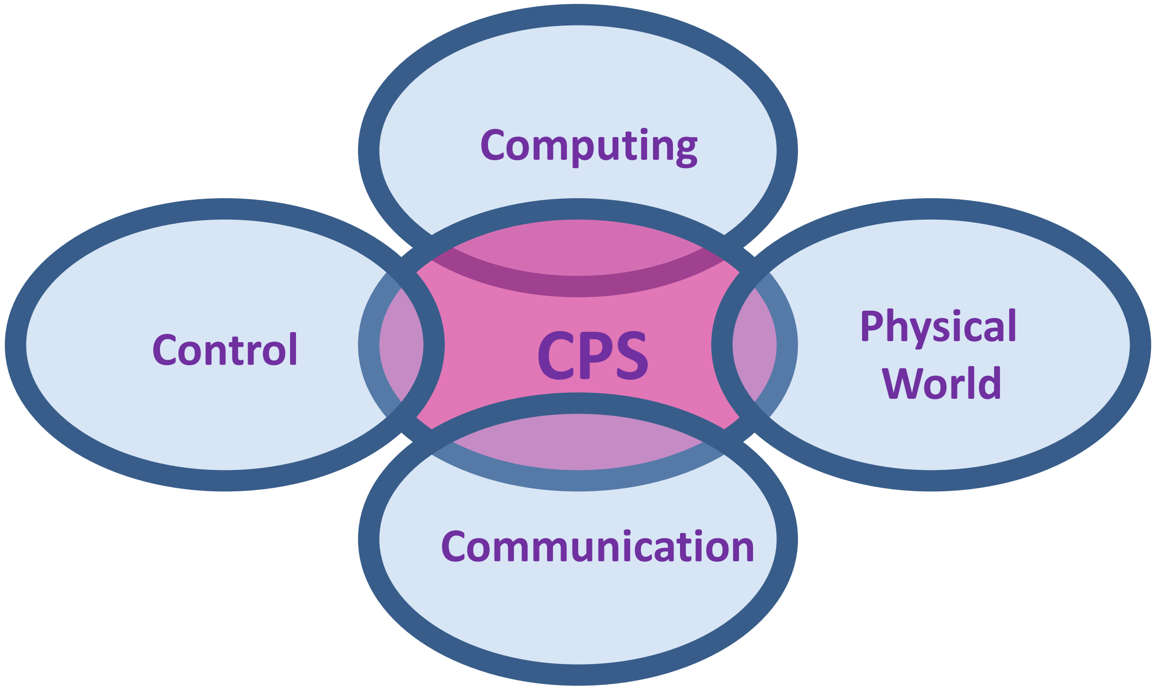


ampions League, Malmö New Stadium
Malmö FF - Real Madrid
nesday 30th of September at 20.45

Content

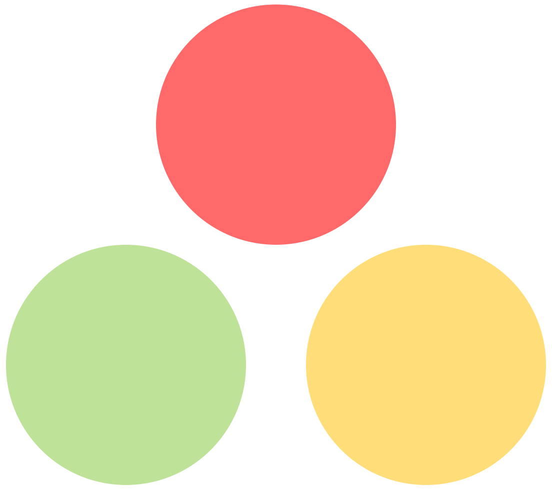
- Cyber-Physical Systems – My Personal View
- Control of Computer Systems
  - Motivation and Background
  - A simple queue length control example
- Resource Management for Multi-Core Embedded Systems
  - The ACTORS Resource Manager
    - Video demo
  - Game-Theory Resource Manager
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications
- A CPS Story

Cyber-Physical Systems



Paradigm Shift

- When designing complex artefacts **separation of concerns** is a good design principle.
  - Buildings
  - Vehicles
  - Distributed Systems

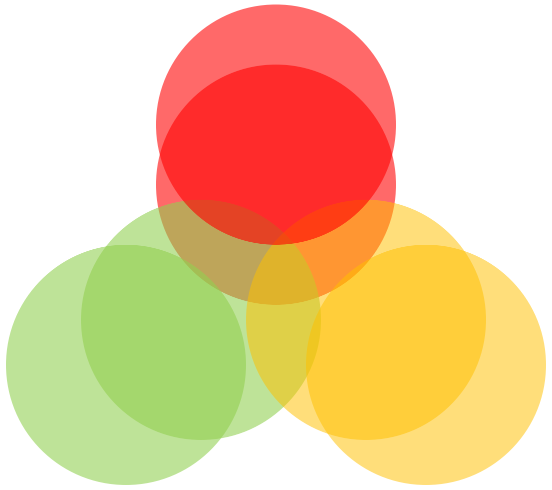


- Obtained through engineering principles, good architectures, design rules etc

- Often the main objective for these design principles is to save **human resources** (engineering time)

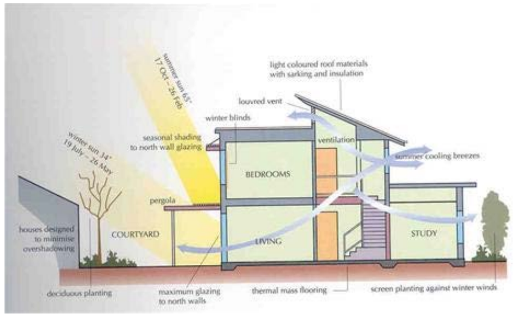
Paradigm Shift

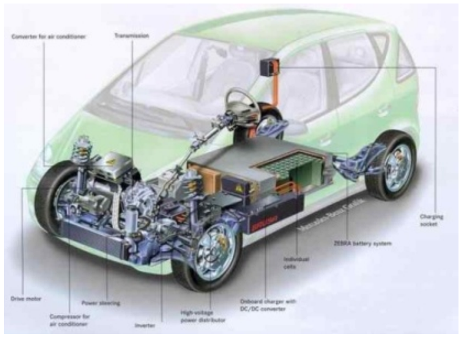
- In many areas the main objective today is to save **natural resources** rather than human
  - Often energy / emissions related
- Crosscutting concerns
  - Traditional separation-based approaches to break down
  - Interaction and interference
- Requires integration of multiple sub-systems both during design and operation
  - Integration-Based Design
  - Codesign
  - Cross-layer design



Some Examples

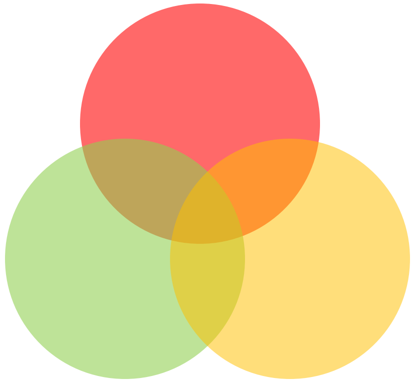
- Smart/green/low-energy buildings
- Green cars
- Server farms
- Battery-driven computing and communication devices
- Cross-layer design and optimization in networks
- Embedded Systems
  - Resource-aware design nothing new!

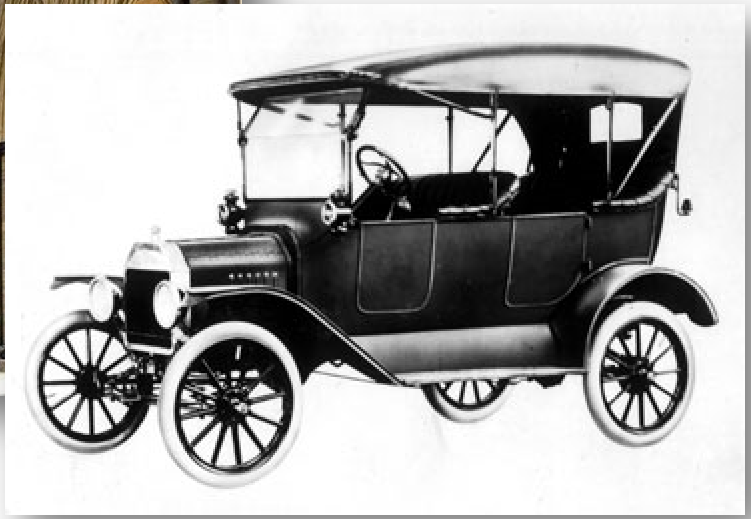

summer sun 65°
17 Oct – 26 Feb
light coloured roof materials with sarking and insulation
louvred vent
winter blinds
seasonal shading to north wall glazing
winter sun 34°
19 July – 26 May
ventilation
summer cooling breezes
BEDROOMS
pergola
houses designed to minimise overshadowing
COURTYARD
LIVING
STUDY
deciduous planting
maximum glazing to north walls
thermal mass flooring
screen planting against winter winds



Paradigm Shift

- Eventually we will find new ways of organizing our work
- Revolutionary paradigm shifts have occ

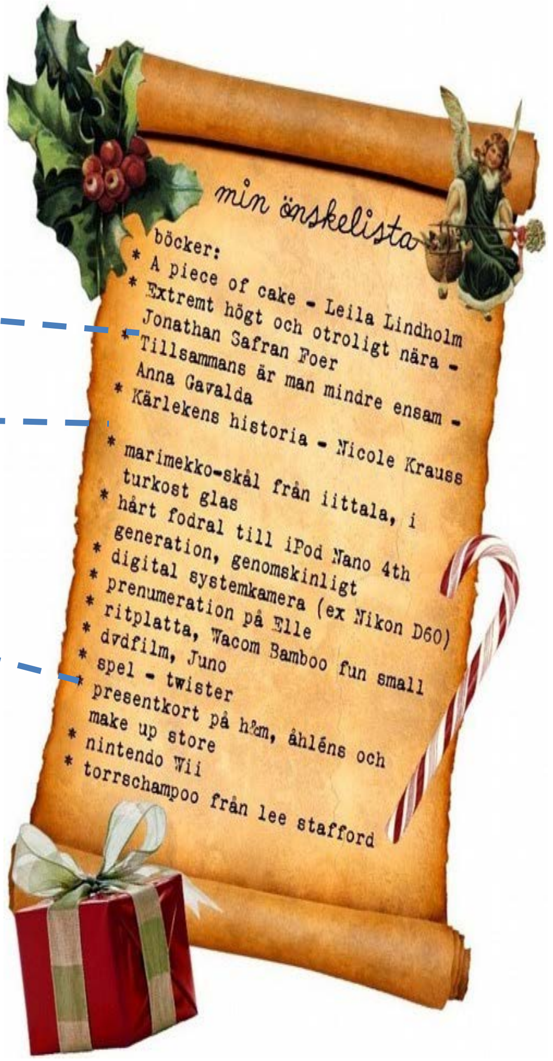




CPS Research Agenda

- Model-Driven Engineering
- Modeling Tools
- Uncertainty Managagement

- Emergent behavior
- Distributed analysis and synthesis of controllers



min önskelista

böcker:

* A piece of cake - Leila Lindholm
* Extremt högt och otroligt nära - Jonathan Safran Foer
* Tillsammans är man mindre ensam - Anna Gavalda
* Kärlekens historia - Nicole Krauss
* marimekko-skål från iittala, i turkost glas
* hårt fodral till iPod Nano 4th generation, genomskinligt
* digital systemkamera (ex Nikon D60)
* prenumeration på Elle
* ritplatta, Wacom Bamboo fun small
* dvdfilm, Juno
* spel - twister
* presentkort på h&m, åhléns och make up store
* nintendo Wii
* torrschampoo från lee stafford

Model-Driven Engineering

- The vision
- Automated process from model to cyber-physical system
  - Functionally correct
  - Dependable
  - Secure
  - Resource-efficient
  - Timeliness
  - …………

Model-Driven Engineering for Software

In the software domain:

- Software abstraction layers tailored for different analysis and design tasks
- Property-preserving model translations and refinement mechanisms
- ”The model is the software”
  - Often realistic
  - Automatic Code Synthesis
- UML

Model-Driven Engineering for HW+SW

In the software + hardware domain:

- SysML + architecture modeling languages such as EAST-ADL, AADL, Modelisar, …
- Allows limited hardware modeling
- Used for modeling the hardware in which the software executes
- Not used for modeling the physical world that the system interact with
  - Too limited behavior models
- Not for CPS

Model-Driven Engineering for CPS

- The model is **not** the system
- Approximations rather than abstractions
  - E.g. reduced models, linearized models, truncated models, …..
- Properties not necessarily maintained
- Models have a limited validity range
- Models expire
  - Requirements change
  - The model and the reality deviate with time due to e.g., aging, wear, ….

Golomb on Modeling

- ”Mathematical Models: Uses and Limitations” – Simulation, Apr 70

➢ Don’t apply a model until you understand the simplifying assumptions on which it is based and can test their applicability.

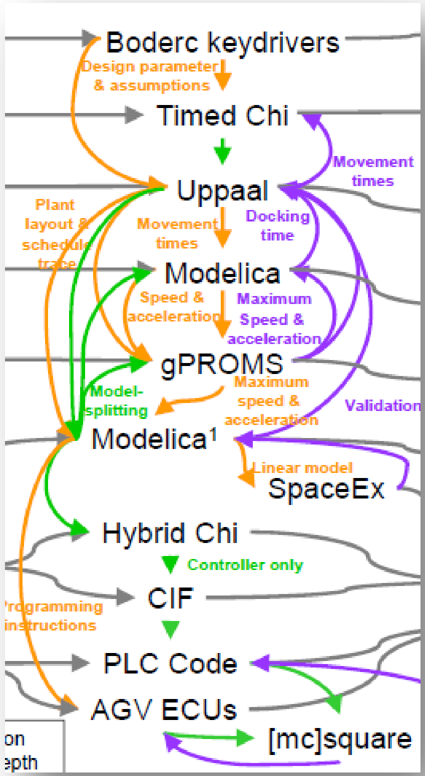
➢ Distinguish at all times between the model and the real world. You will never strike oil by drilling through the map!

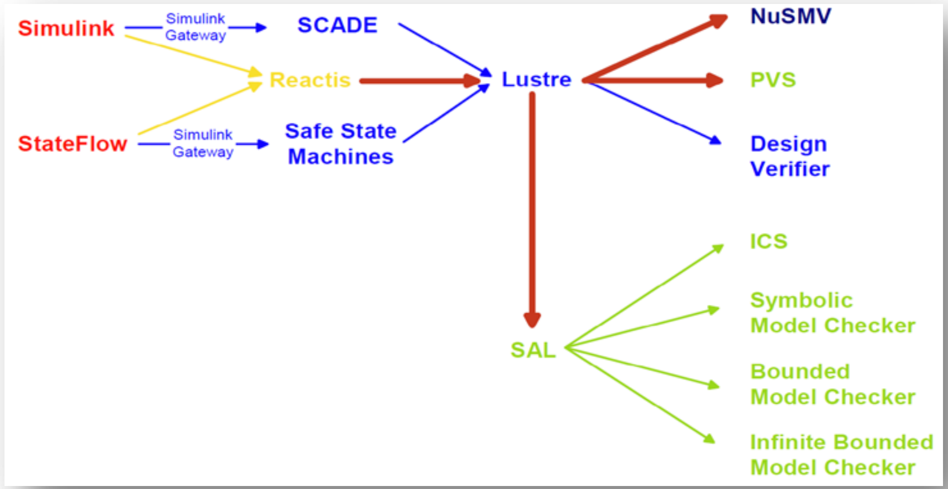
➢ The purpose of notation and terminology should be to enhance insight and facilitate computation – not to impress or confuse the uninitiated

Solomon Wolf Golomb *(1932) mathematician and engineer and a professor of electrical engineering at the University of Southern California.*

Model-Driven Engineering for CPS

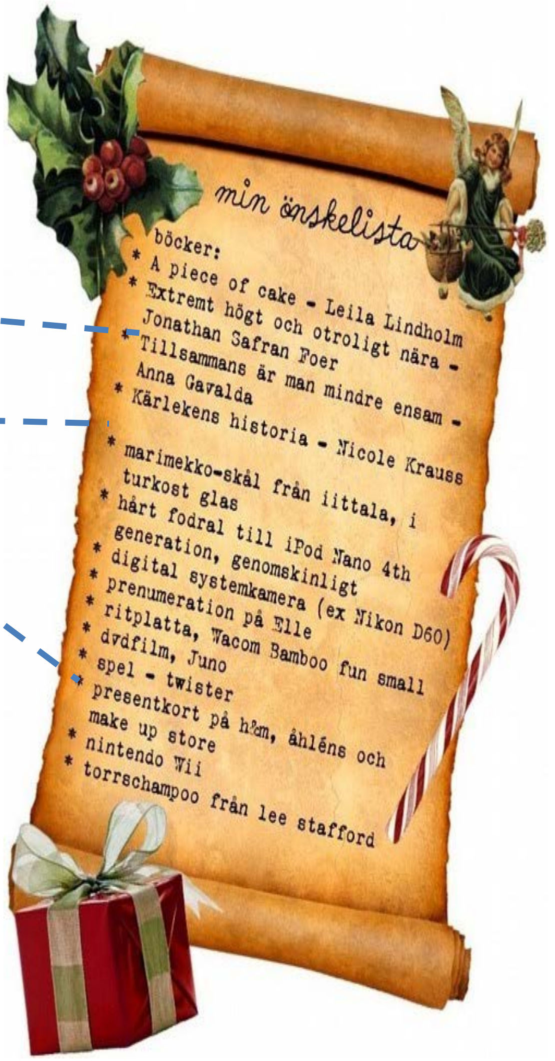
- Most applications of formal methods only concern the discrete-event or discrete-time parts of the CPS (the ”controller” part)
- Long and complicated tool chains





CPS Research Agenda

- Model-Driven Engineering
- Modeling Tools
- Uncertainty Managagement



min önskelista

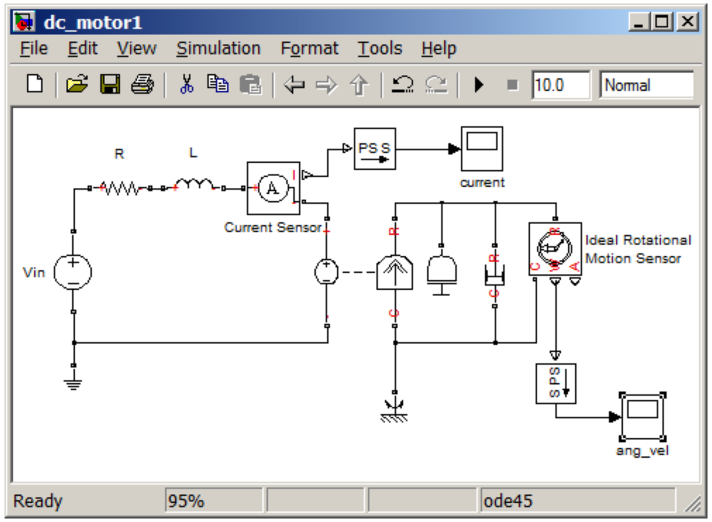
böcker:

* A piece of cake - Leila Lindholm
* Extremt högt och otroligt nära - Jonathan Safran Foer
* Tillsammans är man mindre ensam - Anna Gavalda
* Kärlekens historia - Nicole Krauss

* marimekko-skål från iittala, i turkost glas
* hårt fodral till iPod Nano 4th generation, genomskinligt
* digital systemkamera (ex Nikon D60)
* prenumeration på Elle
* ritplatta, Wacom Bamboo fun small
* dvdfilm, Juno
* spel - twister
* presentkort på h&m, åhléns och make up store
* nintendo Wii
* torrschampoo från lee stafford

Modeling Tools

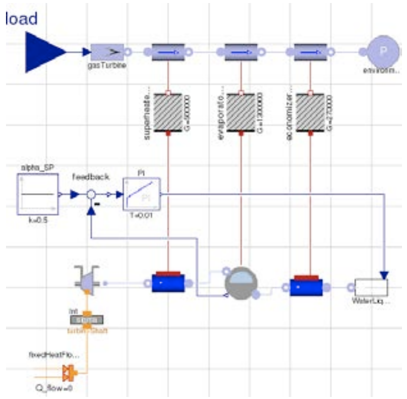
- Frameworks that allow seamless integration of tools
- Or multi-domain/multi-MoC tools
  - Ptolemy II
  - Simulink
    - S-functions allow extensions
      - SimScape → equation-based
      - SimEvents → discrete-event simulation
      - StateFlow → FSM
      - TrueTime → rt kernels + networks

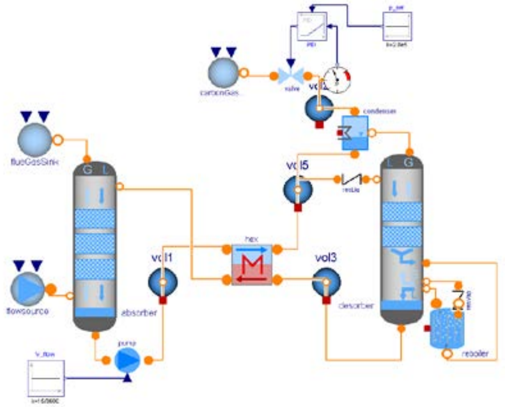

dc_motor1
File Edit View Simulation Format Tools Help
10.0
Normal
R
L
Current Sensor
PS S
current
Vin
Ideal Rotational Motion Sensor
S PS
ang_vel
Ready
95%
ode45

- Equation-based DAE languages have many advantages for the physical parts
  - Modelica, SimScape, Acumen, ..
  - However physical system modeling is difficult
    - Large threshold
    - Problems with high index, initializations, efficient code

Modelica

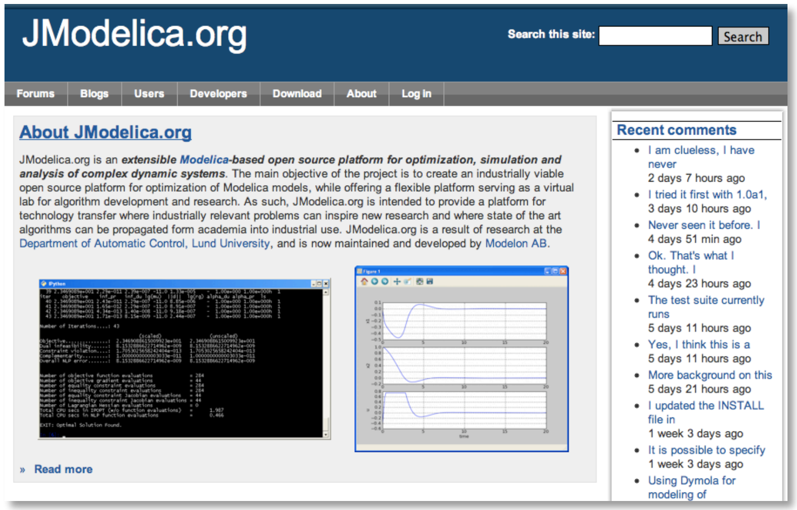
- The most mature equation-based DAE language
- Several commercial tools, e.g. Dymola
- Two open source tools
  - OpenModelica
  - JModelica
- Discrete-time and discrete-event parts designed based on synchronous language ideas
  - Well-defined semantics based on clock inference
- Automatic code generation





JModelica

- Java + C + Python
- JastAdd
  – AspectJ
  – Reference attribute grammars
- Optimica
  – Language extension for representing optimization problems
- Lund University + Modelon



About JModelica.org

JModelica.org is an ***extensible Modelica-based open source platform for optimization, simulation and analysis of complex dynamic systems***. The main objective of the project is to create an industrially viable open source platform for optimization of Modelica models, while offering a flexible platform serving as a virtual lab for algorithm development and research. As such, JModelica.org is intended to provide a platform for technology transfer where industrially relevant problems can inspire new research and where state of the art algorithms can be propagated form academia into industrial use. JModelica.org is a result of research at the Department of Automatic Control, Lund University, and is now maintained and developed by Modelon AB.

```
 39 2.3469089e+001 2.29e-011 2.39e-007 -11.0 1.33e-005    -  1.00e+000 1.00e+000h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
 40 2.3469089e+001 2.43e-011 2.29e-007 -11.0 8.85e-006    -  1.00e+000 1.00e+000h  1
 41 2.3469089e+001 1.65e-012 2.29e-007 -11.0 8.91e-007    -  1.00e+000 1.00e+000h  1
 42 2.3469089e+001 4.34e-013 1.40e-008 -11.0 9.18e-007    -  1.00e+000 1.00e+000h  1
 43 2.3469089e+001 1.71e-013 8.15e-009 -11.0 2.44e-007    -  1.00e+000 1.00e+000h  1

Number of Iterations....: 43

                                   (scaled)                 (unscaled)
Objective...............:   2.3469088615009923e+001    2.3469088615009923e+001
Dual infeasibility......:   8.1532886622714962e-009    8.1532886622714962e-009
Constraint violation....:   1.7053025658242404e-013    1.7053025658242404e-013
Complementarity.........:   1.0000000000003033e-011    1.0000000000003033e-011
Overall NLP error.......:   8.1532886622714962e-009    8.1532886622714962e-009


Number of objective function evaluations             = 284
Number of objective gradient evaluations             = 44
Number of equality constraint evaluations            = 284
Number of inequality constraint evaluations          = 284
Number of equality constraint Jacobian evaluations   = 44
Number of inequality constraint Jacobian evaluations = 44
Number of Lagrangian Hessian evaluations             = 0
Total CPU secs in IPOPT (w/o function evaluations)   =      1.987
Total CPU secs in NLP function evaluations           =      0.466

EXIT: Optimal Solution Found.
```

» Read more

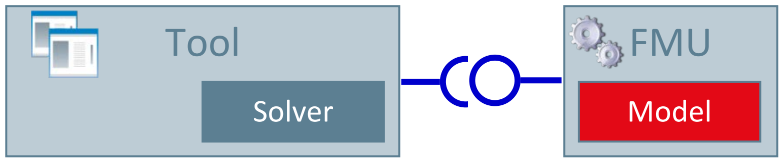
Recent comments

- I am clueless, I have never
  2 days 7 hours ago
- I tried it first with 1.0a1,
  3 days 10 hours ago
- Never seen it before. I
  4 days 51 min ago
- Ok. That's what I thought. I
  4 days 23 hours ago
- The test suite currently runs
  5 days 11 hours ago
- Yes, I think this is a
  5 days 11 hours ago
- More background on this
  5 days 21 hours ago
- I updated the INSTALL file in
  1 week 3 days ago
- It is possible to specify
  1 week 3 days ago
- Using Dymola for modeling of

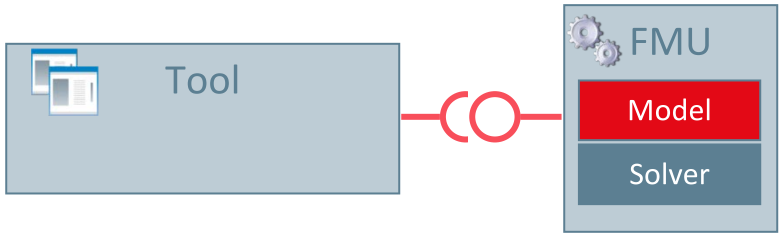
```
optimization VDP_Opt(objective=cost(finalTime),
     startTime=0,
     finalTime(free=true, initialGuess=1))
  VDP vdp(u(free=true,initialGuess=0.0));
   Real cost (start=0);
equation
   der(cost) = 1;
constraint
   vdp.x1(finalTime) = 0;
   vdp.x2(finalTime) = 0;
   vdp.u >= -1; vdp.u <= 1;
end VDP_Opt;
```

Functional Mock-Up Interface (FMI)

- Model exchange standard
  - Open source
  - Non-proprietary
  - cp. S-functions
- Model exchange

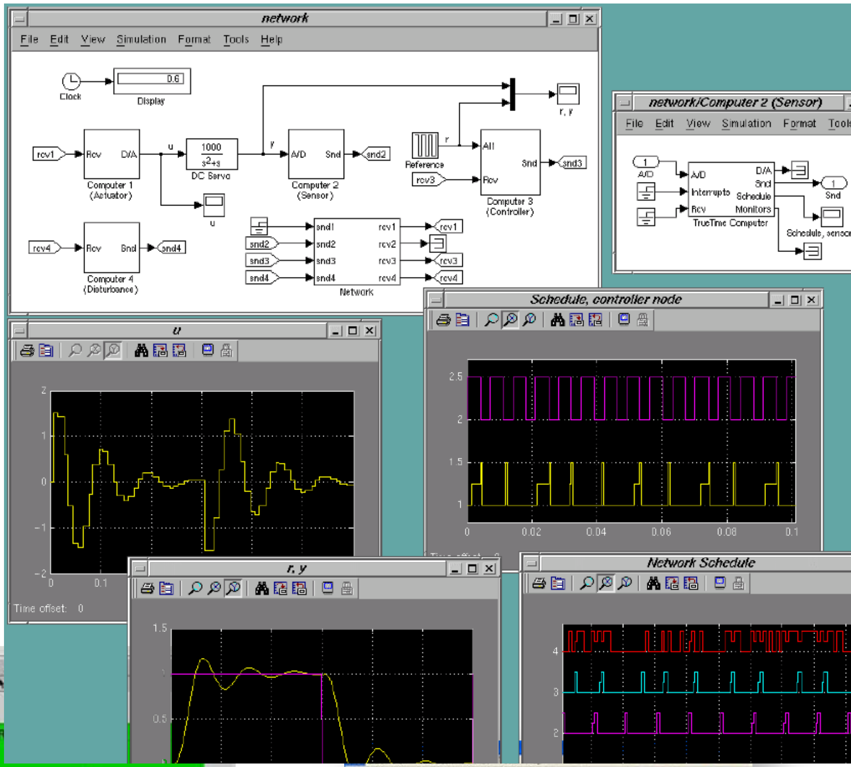


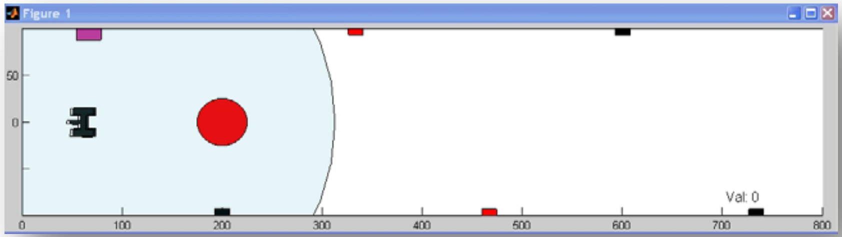
- Co-simulation

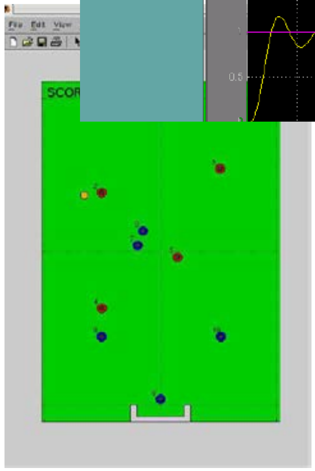


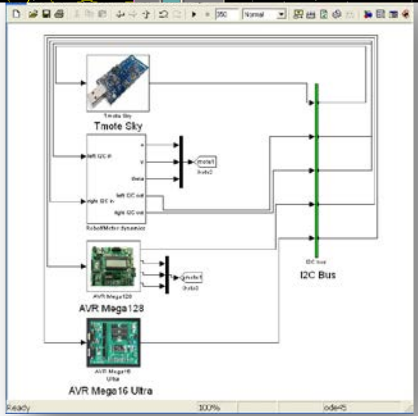
TrueTime

- Co-simulation of controller task execution, network transmissions, and continuous-time plant dynamics
- Simulink using S-functions



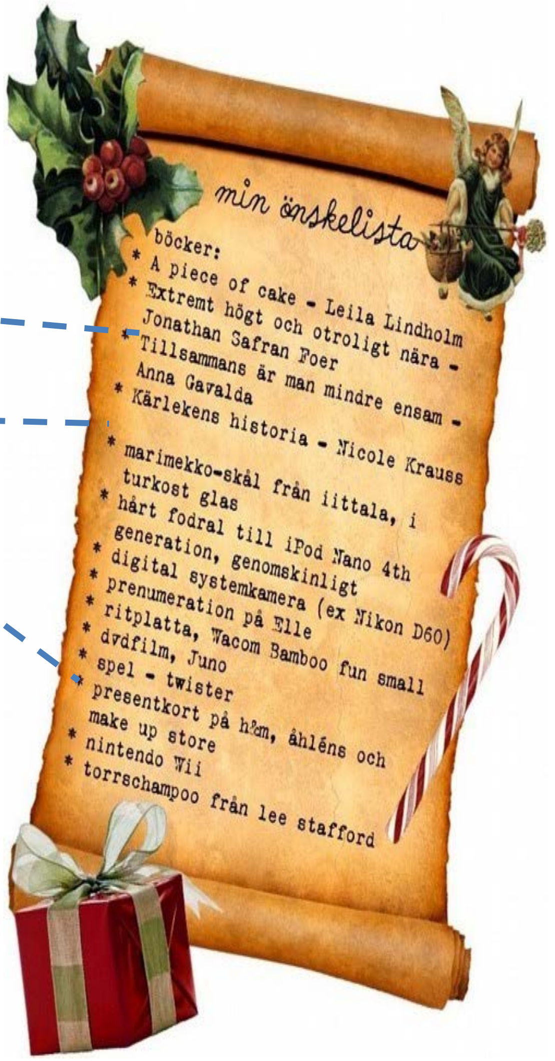






CPS Research Agenda

- Model-Driven Engineering
- Modeling Tools
- Uncertainty Managagement


min önskelista

böcker:

* A piece of cake - Leila Lindholm
* Extremt högt och otroligt nära - Jonathan Safran Foer
* Tillsammans är man mindre ensam - Anna Gavalda
* Kärlekens historia - Nicole Krauss
* marimekko-skål från iittala, i turkost glas
* hårt fodral till iPod Nano 4th generation, genomskinligt
* digital systemkamera (ex Nikon D60)
* prenumeration på Elle
* ritplatta, Wacom Bamboo fun small
* dvdfilm, Juno
* spel - twister
* presentkort på h&m, åhléns och make up store
* nintendo Wii
* torrschampoo från lee stafford

Manage uncertainty

- Resource-sharing in our implementation platforms
  - Cores sharing caches
  - Threads sharing cores
  - Applications sharing computers
  - Communication links sharing
  - media
- Inherent in the physical domain
- Modeling, analysis, simulation, verification, ……

Automatic Control and CPS

1. Implementation of feedback control systems on resource-constrained HW platforms
   – Control and scheduling co-design
   – Temporal robustness (jitter and delays)
   – Event-based control and Self-triggered control
   – Will not talk about this
2. Control of computer systems
   – The main topic of my lecture

Content

- Cyber-Physical Systems – My Personal View
- **Control of Computer Systems**
  - **Motivation and Background**
  - A simple queue length control example
- Resource Management for Multi-Core Embedded Systems
  - The ACTORS Resource Manager
    - Video demo
  - Game-Theory Resource Manager
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications
- A CPS Story

Control of Computer Systems

- Apply control as a techniques to manage uncertainty and achieve performance and robustness in computer and communication systems.
- Applications in
  - Internet
  - Servers and data centers, i.e., the cloud
  - Cellular phone systems
  - Embedded systems

Control of Computer Systems

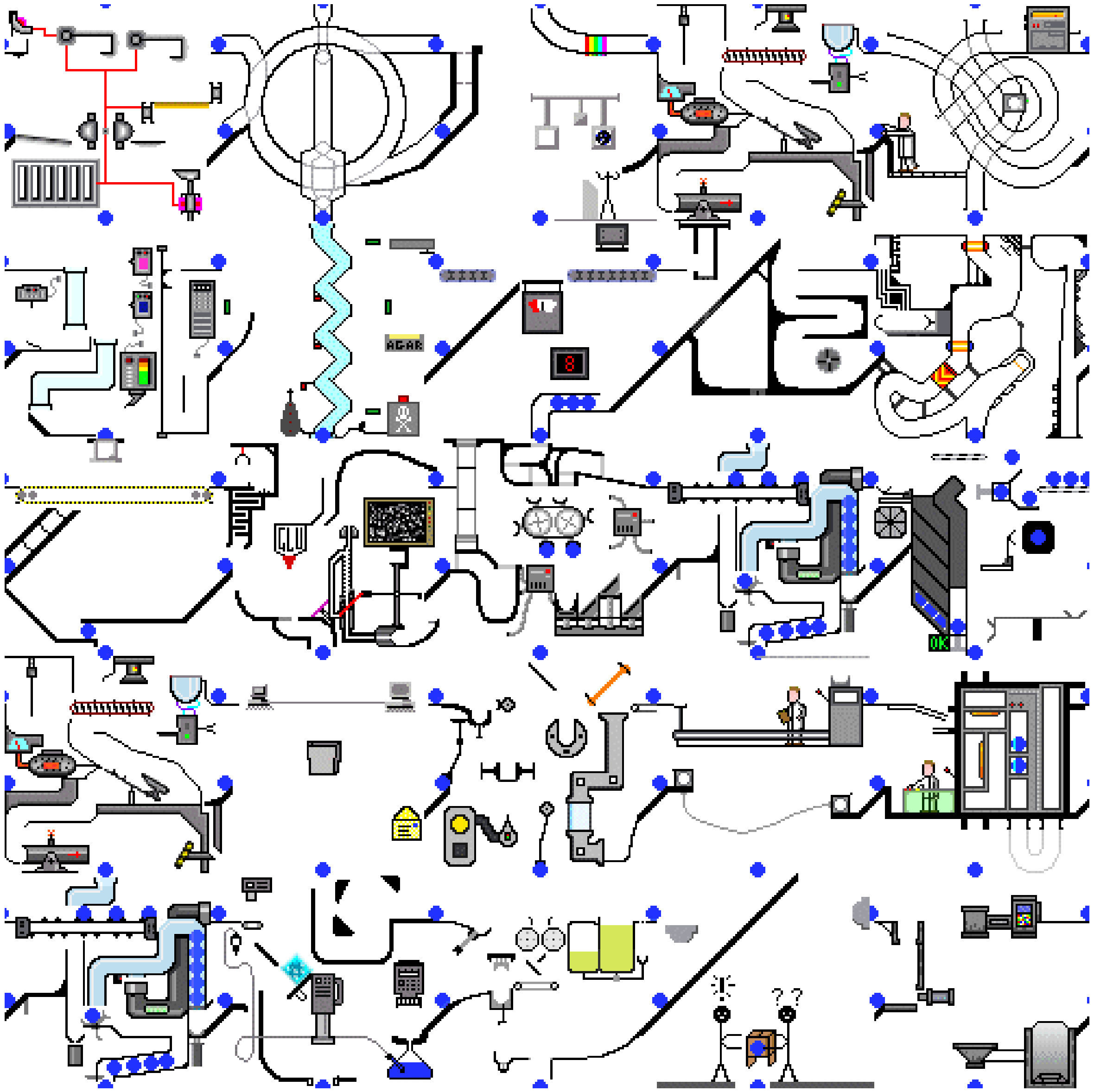
Alternative names:

- Dynamic/adaptive resource management
  - Control as means for managing limited resources
  - Adaptivity from a CS point of view
- Feedback computing/scheduling
- Autonomous/autonomic computing
- Reconfigurable computing

Why?

- System complexity increases

Complexity



Why?

- System complexity increases
- Complete information about all use cases and their resource requirements is often not available at design-time
- Green computing → power consumtion constraints increasingly important
- Increased hardware density → thermal constraints increasingly important
- Hardware platforms increasingly complex → increasing difficulties in providing good off-line estimates of resource consumption
- Hardware variability increases
- Increased requirements on dependability
- Hardware increasingly often support adaptivity
- Increased requirements on predictability in the cloud

Control of Computer Systems

- Active research area since around 2000
- However, feedback has been applied in ad hoc ways for long without always understanding that it is control, e.g. TCP/IP
- Control of computing systems can benefit from a lot of the classical control results
  - However, several new challenges
  - First principles-based modeling not so natural
  - Complex dynamics no longer the problem

Some Examples

Example 1: A multi-mode embedded system where the resource requirements for all the tasks in all the modes are known at design time

- Use schedulability analysis to ensure that the deadlines are met in all modes and then use a mode-change protocol that ensures that all deadline also are met during the transition between the modes

Example 2: An embedded system with a constant set of hard-RT applications/tasks but where the WCET analysis possible on the selected hardware is too pessimistic and leads to too low resource utilization or where the age- or process-induced variability is too large

- Measure the actual resource consumption and adjust, e.g. the task rates in order ensure that the schedulability condition is fulfilled

Some Examples

Example 3: Open embedded systems where the number of applications and their characteristics change dynamically (e.g, smartphones)

- Measure resource consumption and decide how much resources that should be allocated to each application in order to maximize QoS/QoE while minimizing power consumption and avoiding thermal hotspots

Example 4: A distributed embedded system where one for dependability reasons must be able to ensure system functionality also in case of single-node failures

- Detect node failures and then adapt the task mapping and the schedules so that the system performance is still acceptable

Some Examples

Example 5: An FPGA-based system with multiple modes that is too large to fit in a single FPGA or where the power consumption will be too high

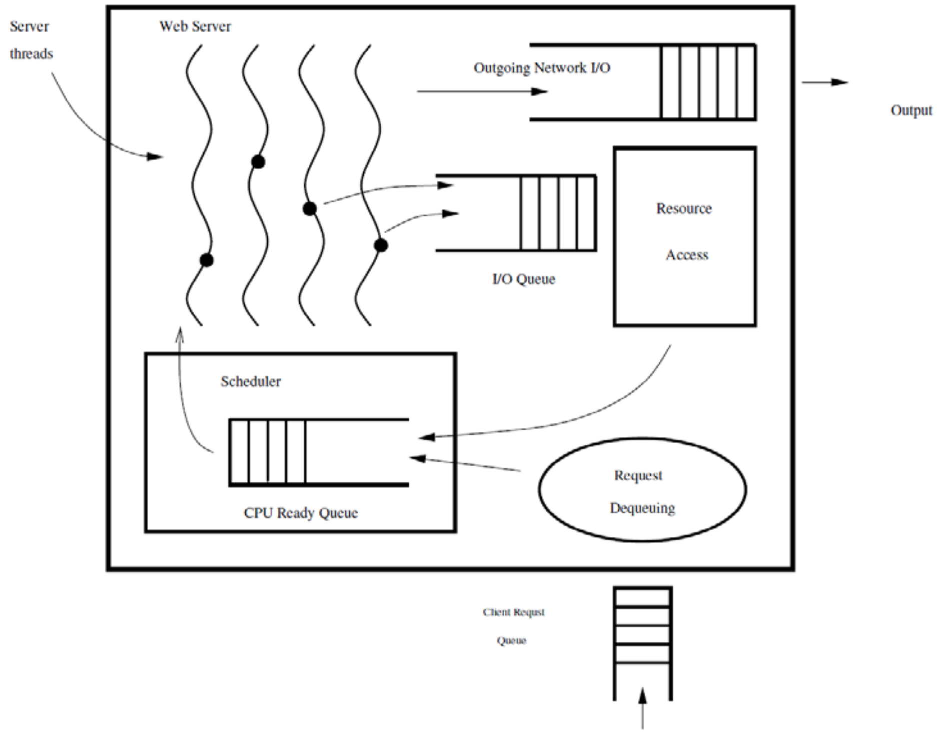
- Use run-time reconfiguration to change the FPGA function dynamically

Example 6: A cloud deployed web-service application where the incoming load varies a lot over time

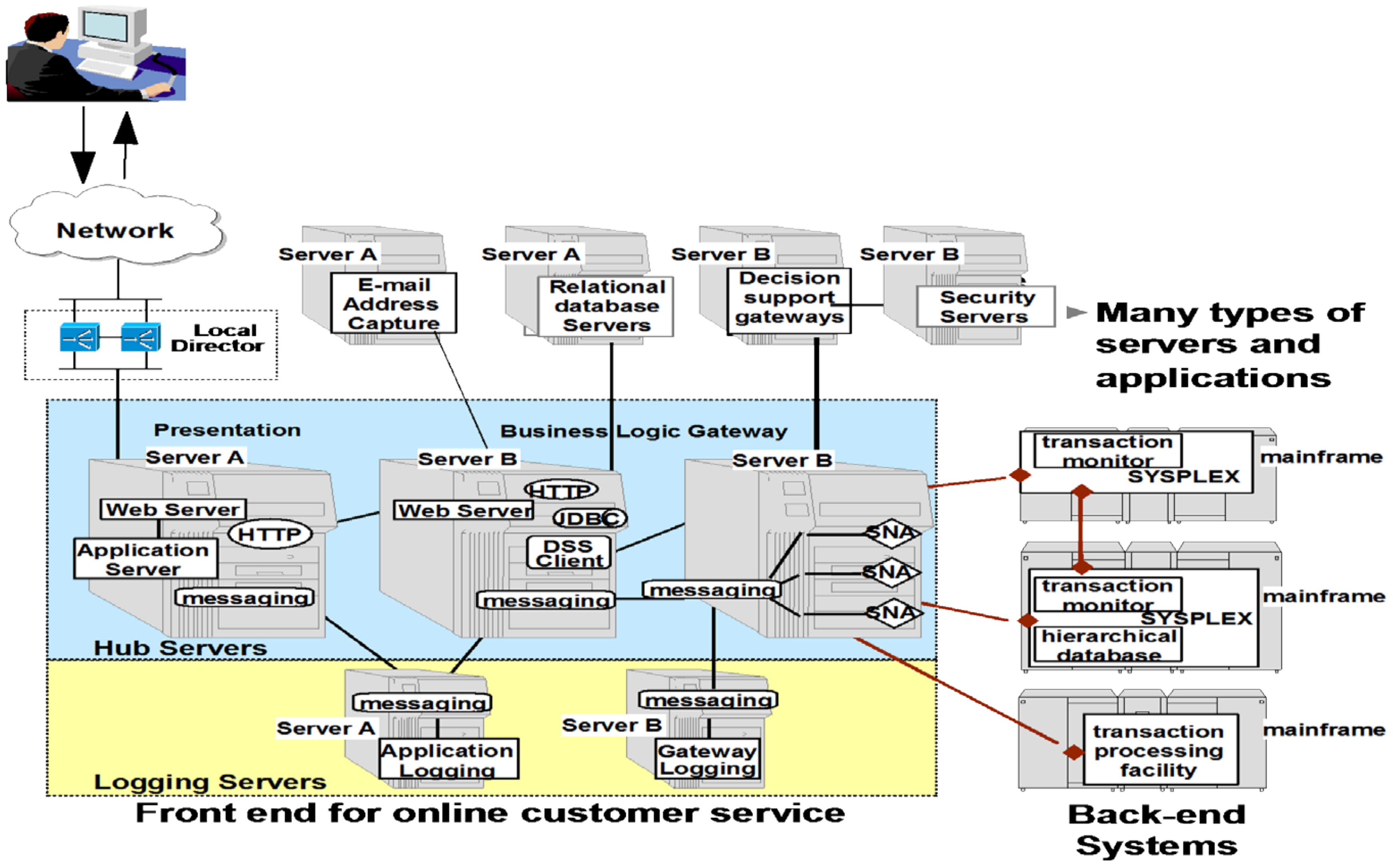
- Dynamically add or remove virtual machines to match the load (elasticity control/auto-scaling)

Computer Internals

- Execution/service times
- Queuing delays
- Discrete Event Dynamic System
  - Tasks/requests arrive (queued) and depart (dequeued)



(Cloud) Server Systems



Content

- Cyber-Physical Systems – My Personal View
- **Control of Computer Systems**
  - Motivation and Background
  - **A simple queue length control example**
- Resource Management for Multi-Core Embedded Systems
  - The ACTORS Resource Manager
    - Video demo
  - Game-Theory Resource Manager
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications
- A CPS Story

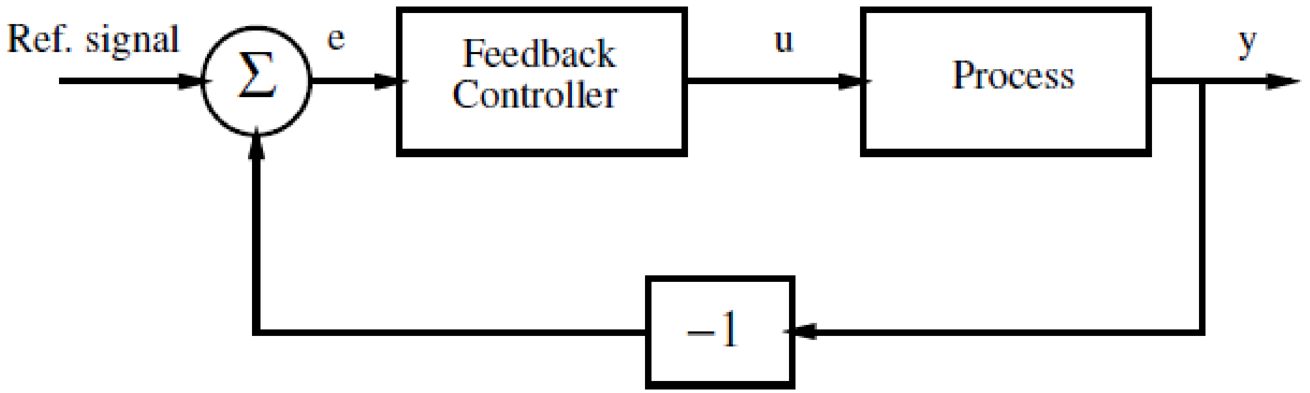
Modeling and Control Formalisms

- Discrete Event Formalisms
  - Automata theory (e.g., Supervisory Control Theory)
  - Petri nets
  - Often problem with scalability
  - Queuing theory
- Continuous-Time Formalisms
  - Liquid (”flow”) models + continuous-time control
  - Queues = tanks, computations = flows
  - Average values assuming large number of requests/jobs
  - Sometimes event-driven sampling and control

The Feedback Principle

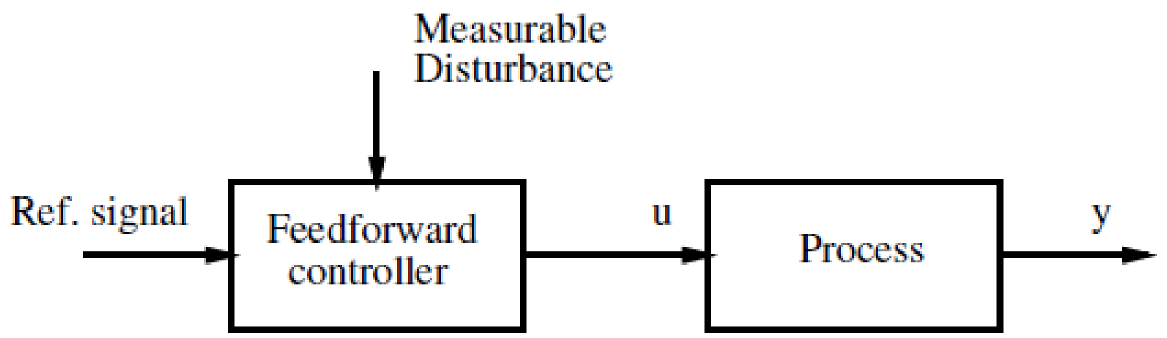
A very powerful idea, that often leads to revolutionary changes in the way systems are designed.

The primary paradigm in automatic control.



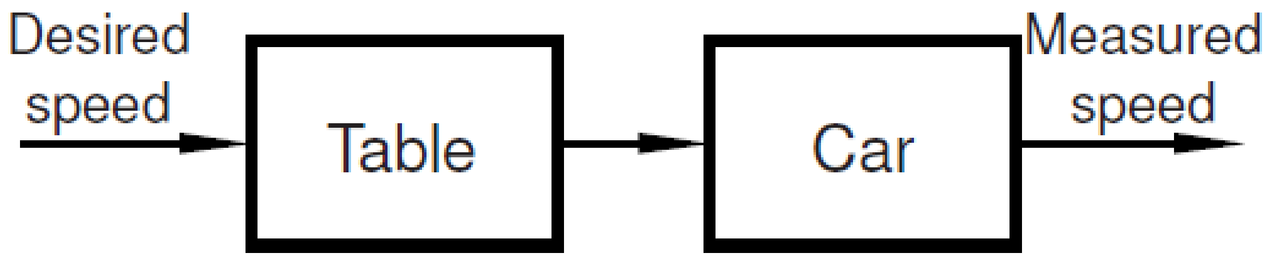
- Base corrective action on an error that has occurred
- Closed loop

The Feedforward Principle



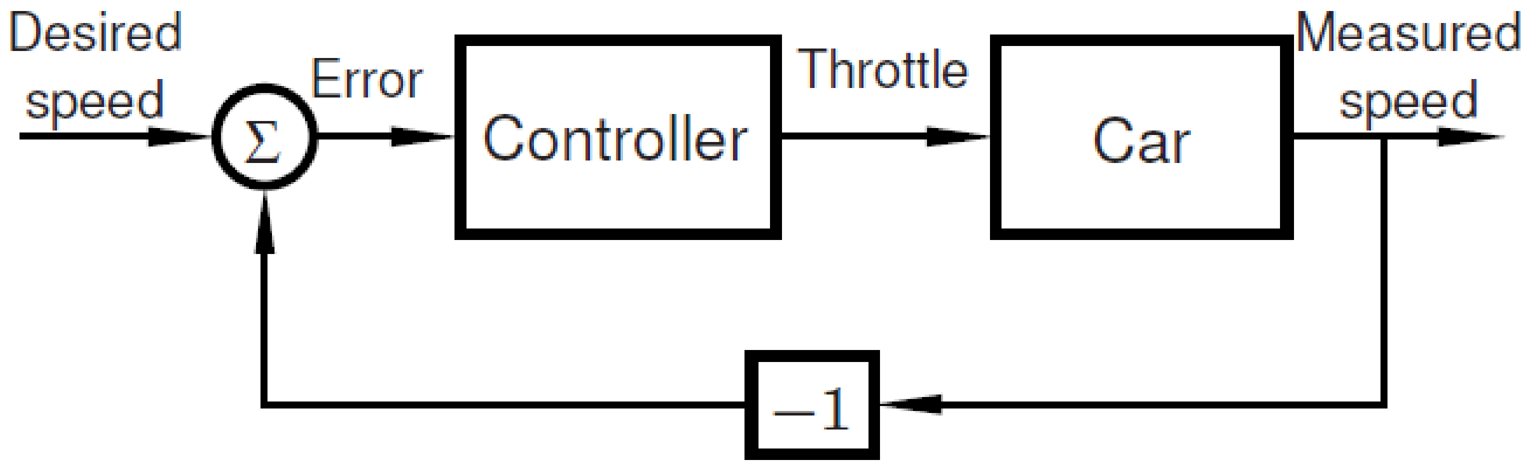
- Take corrective action before an error has occurred
- Measure the disturbance and compensate for it
- Use the fact that the reference signal is known and adjust the control signal to the reference signal
- Open loop

Example: Feedforward Based Cruise Controller



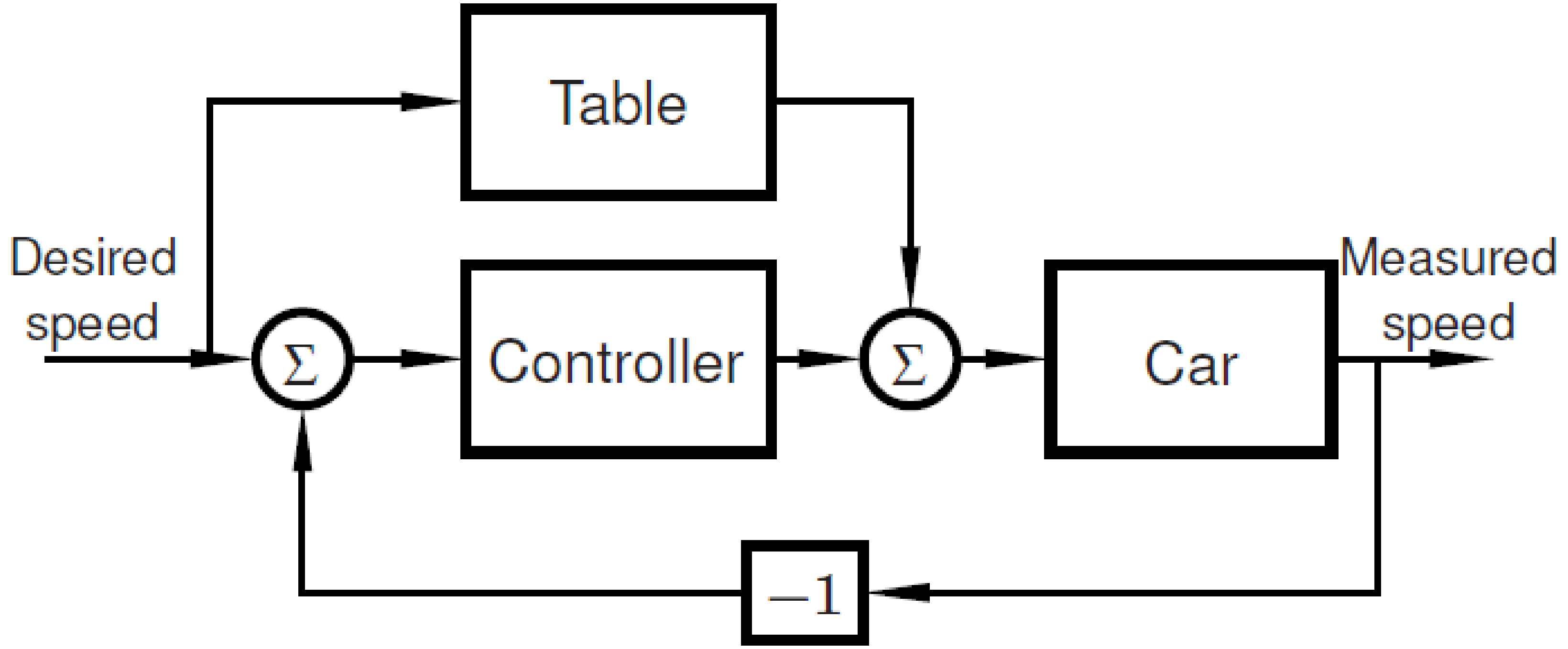
- Open loop
- Problems?

Example: Feedback-Based Cruise Controller



- Closed loop
- Simple controller:
  - Error > 0: increase throttle
  - Error < 0: decrease throttle

Example: Feedback + Feedforward Based Cruise Controller



- Both proactive and reactive

Common Feedback Controllers

- Proportional Controller (P)

$$u(t) = Ke(t)$$

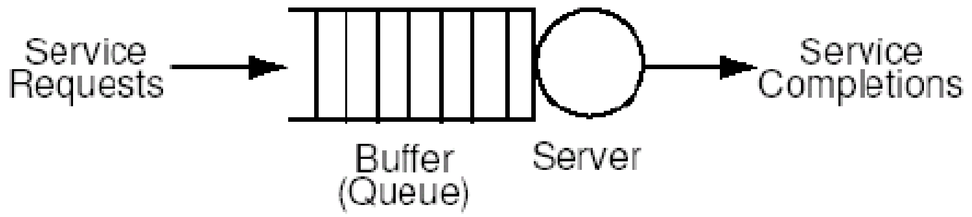
- Proportional and Integral Controller (PI)

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(s)ds\right)$$

- Proportional, Integral and Derivative Controller (PID)

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(s)ds + T_d\frac{de(t)}{dt}\right)$$

Example: Queuing System



Work requests (customers) arrive and are buffered

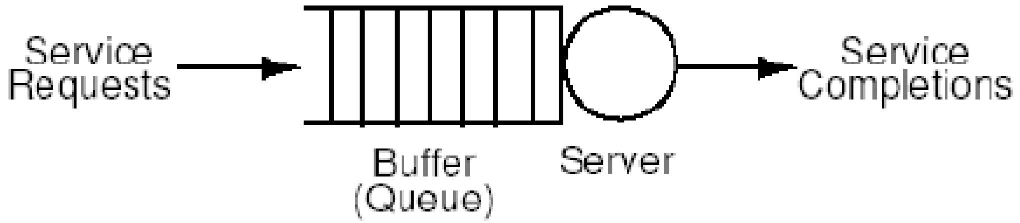
Service level objectives (e.g., response time for request belonging to class X should be less than Y time units)

Reduce the delay caused by other requests, i.e., adjust the buffer size and redirect or block other requests

Admission control

Example: Queue Length Control

Assume an M/M/1 queuing system:

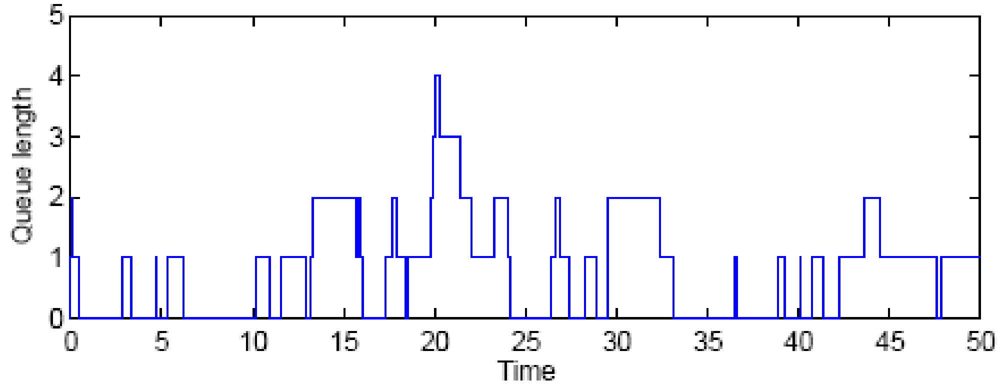


- Random arrivals (requests), Poisson distributed with average λ per second
- Random service times, exponentially distributed with average $1/\mu$
- Queue containing x requests

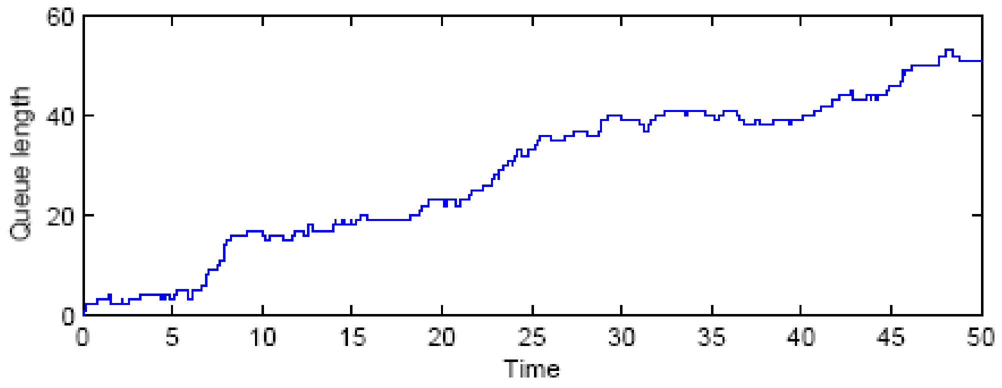
Intuition: $x \to \infty$ if $\lambda > \mu$

Queue Length Control: Simulation

$\lambda = 0.5,\ \mu = 1$:



$\lambda = 2.0,\ \mu = 1$:



Queue Length Control: Model

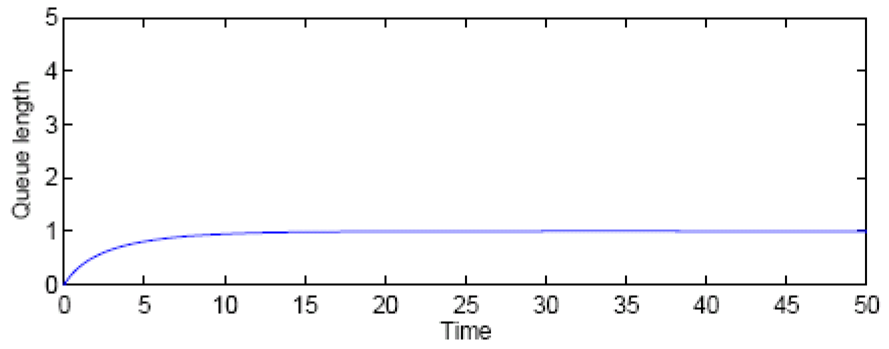
Approximate the system with a nonlinear flow model (Tipper's model from queuing theory)

The expectation of the future queue length x is given by

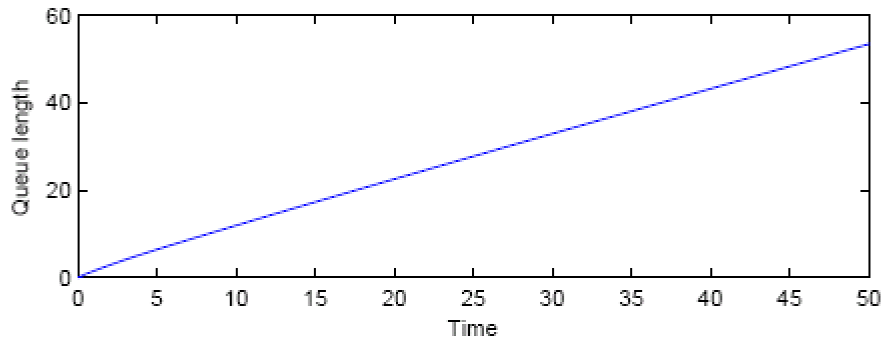
$$\dot{x} = \lambda - \mu \frac{x}{x+1}$$

Queue Length Control: Model

$\lambda = 0.5,\ \mu = 1$:



$\lambda = 2.0,\ \mu = 1$:



Queue Length Control: Control Signal

Control the queue length by only admitting a fraction u (between 0 and 1) of the requests

$$\dot{x} = \lambda u - \mu \frac{x}{x+1}$$

Admission control

Queue Length Control: Linearization

Linearize around $x = x^\circ$

Let $y = x - x^\circ$

$$\dot{y} = \lambda y - \mu \frac{1}{(x^\circ + 1)^2} y = \lambda u - \mu a y$$

Queue Length Control: P-control

$u = K(r - y)$

$$\dot{y} = \lambda K(r - y) - \mu a y$$

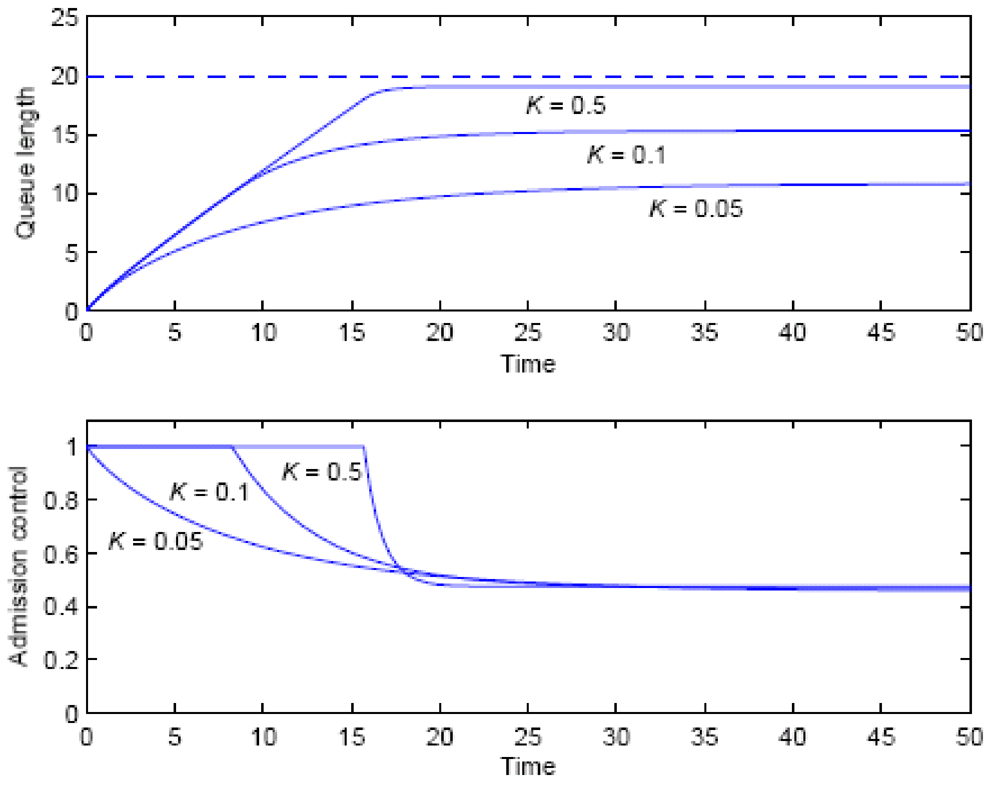
$$(s + \lambda K + \mu a)Y(s) = \lambda K R(s)$$

$$G_{cl}(s) = \frac{\lambda K}{s + \lambda K + \mu a}$$

With K the closed loop pole can be placed arbitrarily

Queue Length Control: P-Control

Simulations for $\lambda = 2, \mu = 1, x^\circ = 20$ and different values of K



Queue Length Control: PI-control

$$G_P(s) = \frac{\lambda}{s + \mu a}$$

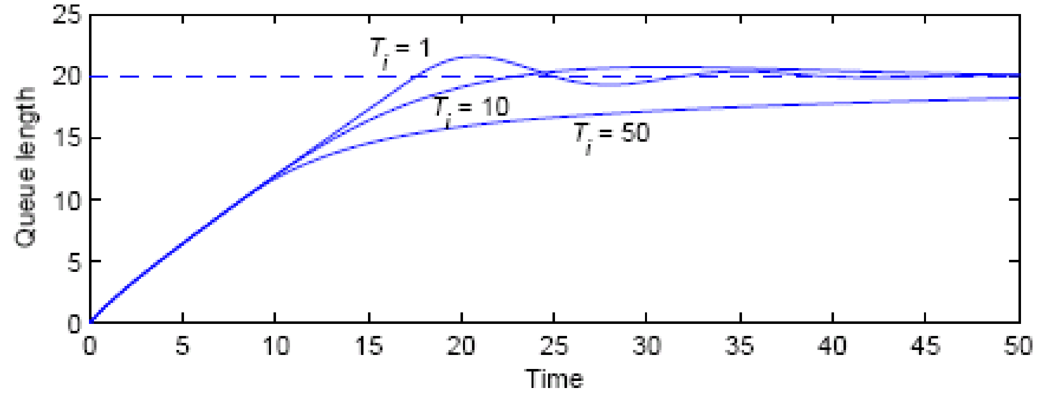
$$G_R(s) = K(1 + \frac{1}{sT_i})$$

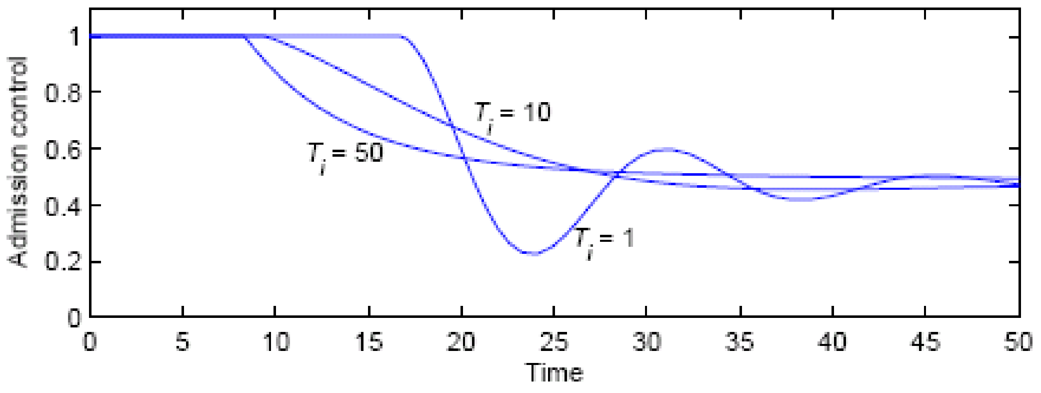
$$G_{cl}(s) = \frac{G_P G_R}{1 + G_P G_R} = \frac{\lambda K(s + \frac{1}{T_i})}{s(s + \mu a) + \lambda K(s + \frac{1}{T_i})}$$

With K and T_i the closed loop poles can be placed arbitrarily

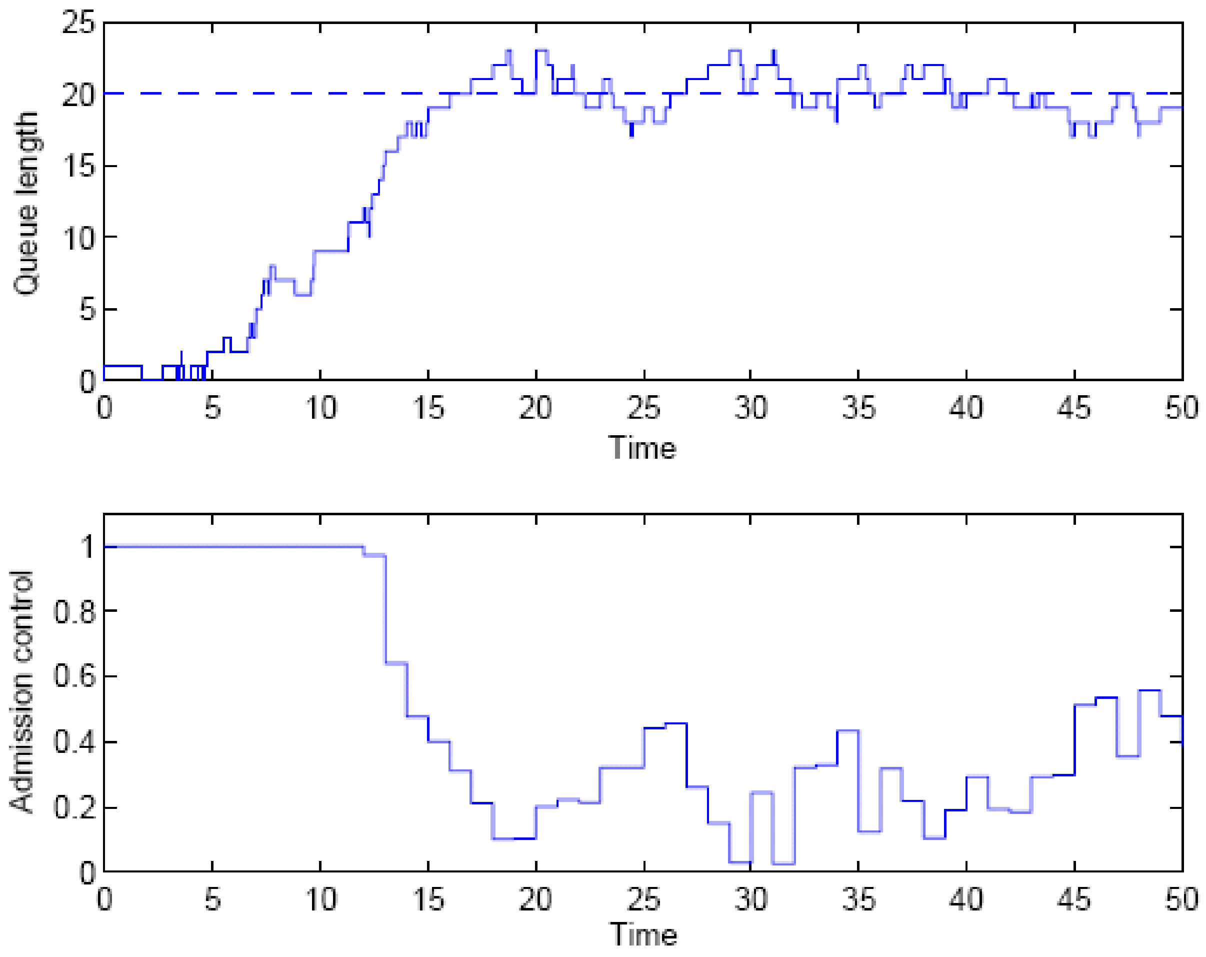
Queue Length Control: PI-control

Simulations for $\lambda = 2, \mu = 1, x^\circ = 20, K = 0.1$ and different values of T_i



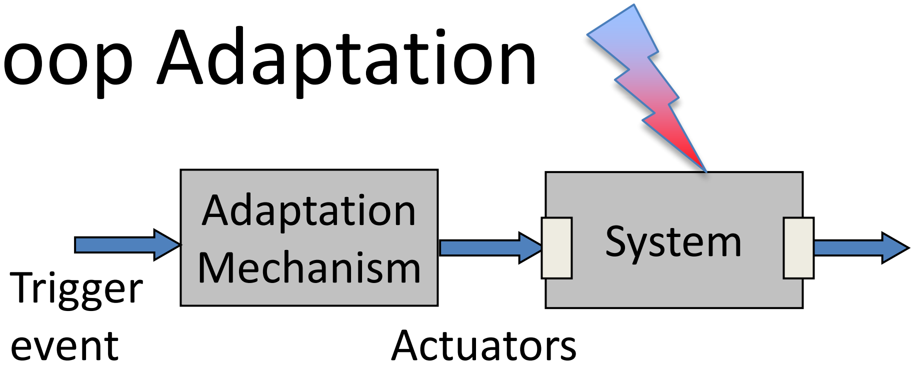


PI-control on Real Queue



Adaptation Mechanisms

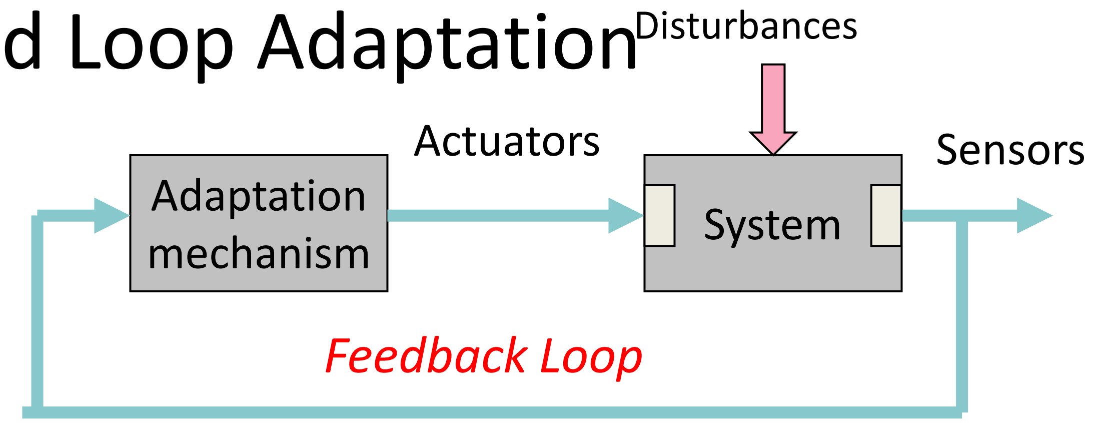
- Open Loop Adaptation



  - Feedforward
  - Assumes perfect information (model) of the system
  - Assumes that there are no external disturbances

Adaptation Mechanisms

- Closed Loop Adaptation



  - Feedback
  - Adaptation Mechanism == Controller
  - Requires sensors
  - May cause unstabilities

Adaptation Formulations

- Often formulated as an optimization-problem or as a control-problem
- Optimization Formulations:

maximize/minimize resource-consumption objective
s.t. perfomance constraint

maximize/minimize performance objective
s.t. resource consumption constraint

  - Performed off-line, online when some change has occurred or periodically, off-line+on-line, …
  - ILP, Bin-packing, MILP, QP, NLP (B&B, GA, CP …)
  - Centralized or distributed

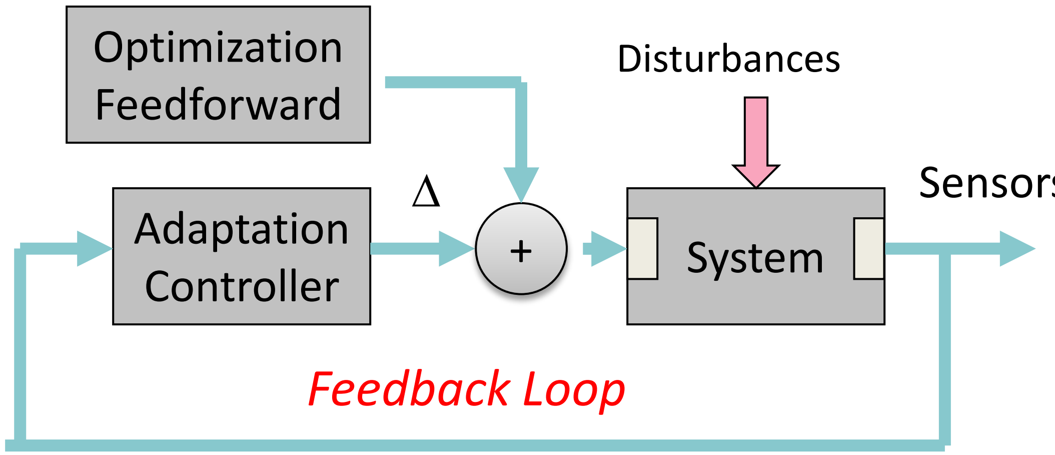
Adaptation Formulations

- Control Formulations:
  - System modelled as (linear) dynamic system
  - Classical linear control design techniques
    - PID
    - LQG
    - ….
  - Designed to obtain a stable closed loop system with desired dynamic performance

$$\boxed{u(t) = K\left(e(t) + \frac{1}{T_I}\int e(s)ds + T_D\frac{de(t)}{dt}\right)}$$

Adaptation Formulations

- Combined Optimization and Control Formulations:
  - Model-Predictive Control (MPC)
    - Optimization problem solved each sample
    - Only the first control signal is used (receding horizon principle)
    - Optimization problem ban be solved off-line (explicit MPC / multiparametric programming) → piecewise affine mapping
  - Feedforward + feedback structures



Actuators

- Change the applications / threads
  - For example:
    - Accept or reject decision
    - Change the rates of periodic processes
    - Change between alternative versions (service/quality levels)
    - Anytime formulations
  - Often requires support from the applications
- Change the mapping of the application onto the execution platform
  - Priority
  - Schedule
  - Processor allocation

Actuators

- Change the execution platform
  - Number of processors (virtual or physical)
    - DPM techniques
  - Speed of processors
    - DVFS
    - Change the bandwidth of the VM or bandwidth server
  - Functionality (hardware-based systems)
    - Micro-code in soft-cores
    - FPGA netlist

Sensors

- What we can (or would we like to) measure?
  - Application performance
    - Obtained QoS
    - Throughput
    - Latency
  - OS / CPU level
    - CPU cycles / task
    - CPU utilization
    - Deadline miss ratio
  - Power and temperature
    - Power consumption for each unit
    - Temperature of each heat source (core, coprocessor, memory controller, ….)

Problems of Feedback

Feedback can introduce new problems:

- The feedback mechanism itself consumes resources
- Harder to provide formal guarantees about the system → not suitable for safety-critical hard real-time application, or?

What about Safety-Critical Systems?

- In many cases control systems
- Due to the feedback errors in the space domain are natural
- Control system designed using
  - Numerous approximations
    - Model reduction, linearization, ...
  - Verified through extensive simulations
  - Large safety margins when selecting, e.g., sampling periods
- Why is it then so unthinkable to use feedback also at the implementation level?

Problems of Feedback

Feedback can introduce new problems:

- The feedback mechanism itself consumes resources
- Harder to provide formal guarantees about the system → not suitable for safety-critical hard real-time application, or?
- Adds to the complexity
- May complicate the design process (modeling, V&V, …)
- Requires tuning
- Sensors and actuators are necessary
- Models are necessary
  - Of the system
  - Of the feedback mechanism itself
- Feedback may cause instability
  - In my mind stability is much overrated

Problems of Feedback

Feedback can introduce new problems:

- The feedback mechanism itself consumes resources
- Harder to provide formal guarantees about the system → not suitable for safety-critical hard real-time application, or?
- Adds to the complexity
- May complicate the design process (modeling, V&V, …)
- Requires tuning
- Sensors and actuators are necessary
- Models are necessary
  - Of the system
  - Of the feedback mechanism itself
- Feedback may cause instability
  - In my mind stability is much overrated
- Feedback may introduce measurement noise
  - Only when you measure physical entities!

Content

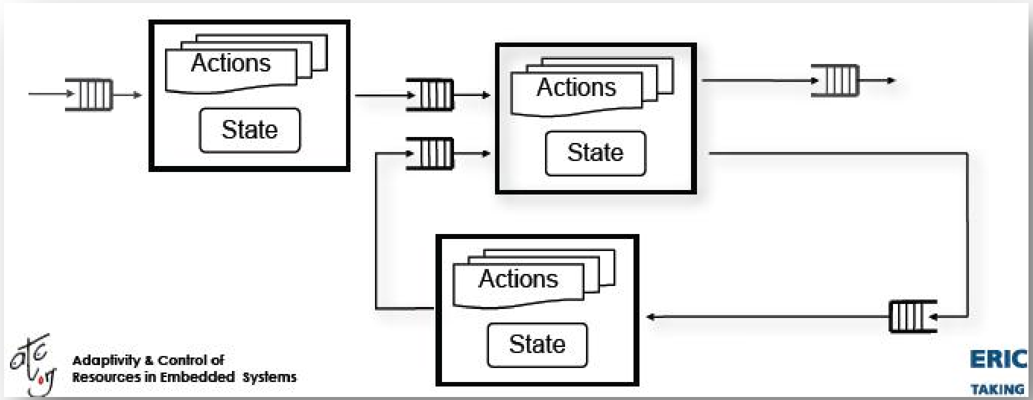
- Cyber-Physical Systems – My Personal View
- Control of Computer Systems
  - Motivation and Background
  - A simple queue length control example
- **Resource Management for Multi-Core Embedded Systems**
  - **The ACTORS Resource Manager**
    - **Video demo**
  - Game-Theory Resource Manager
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications
- A CPS Story

ACTORS

- Adaptivity and Control of $\overset{\text{CPU}}{\vee}$ Resources in Embedded Systems
- EU FP7 STREP project
  - 2008-2011
  - Coordinated by Ericsson (Johan Eker)
  - Lund University, TU Kaiserslautern, Scuola Superiore Sant’Anna di Pisa, EPFL, AKAtech, Evidence
- Media applications (soft real-time) for smart phones
- Control applications

ACTORS: Key Ingredients

1. Data-Flow Programming
   – CAL Actor Language



2. Adaptive Resource Management of service-level aware applications
   – Soft real-time media applications
   – Control applications

Service Level-Aware Applications

- Application knob
  - Decides the QoS achieved and the amount resources required
  - High SL → high QoS & high resource usage
  - Low SL → low QoS & low resource usage
- Discrete
  - ”application modes” – the case in ACTORS
- Continuous
  - e.g., sampling rate in a controller

Service level example

SL1: 640x480
CPU: 30%

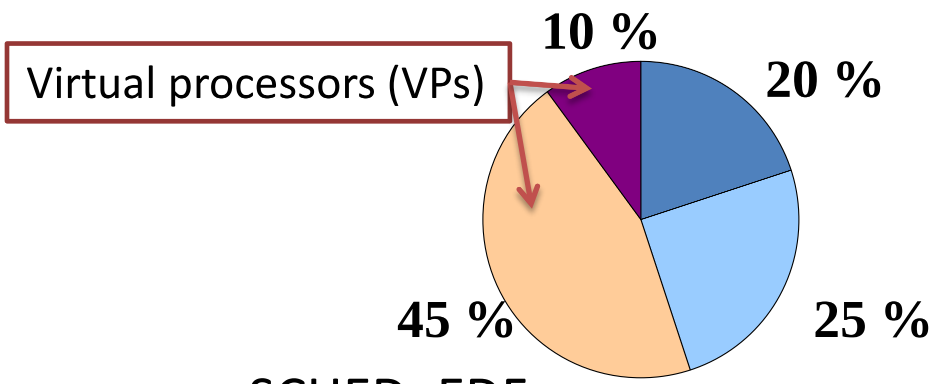
SL2: 800x600
CPU: 60%

SL3: 1024x768
CPU: 90%

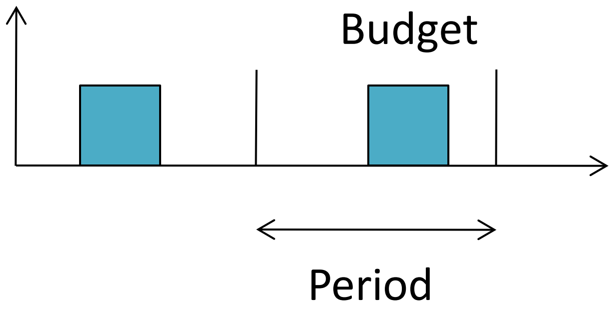
- SL: Resolution and/or frame rate of a video stream
- QoS and required CPU for encoding and decoding depends on the SL

ACTORS: Key Ingredients

3. Reservation-Based CPU Scheduling



- Periodic Bandwidth Servers
  - Constant Bandwidth Server

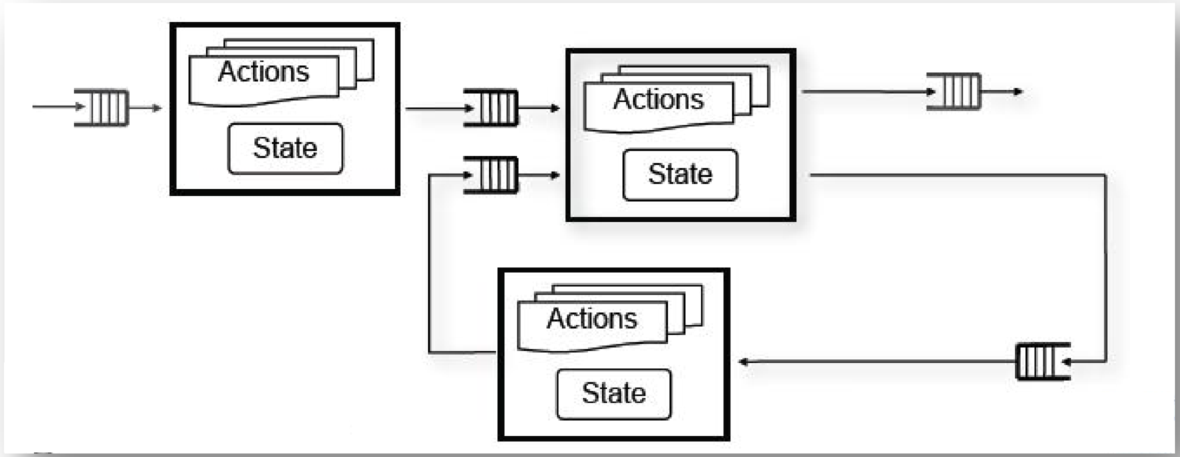


   – SCHED_EDF
     - Partitioned multi-core EDF scheduler
     - Hard CBS reservations
     - Each reservation may contain several threads
     - Hierarchical scheduler with SCHED_EDF tasks executing on a higher level than ordinary Linux tasks

4. Multicore Linux Platforms
   – ARM 11, x86

ACTORS: Dataflow Modeling

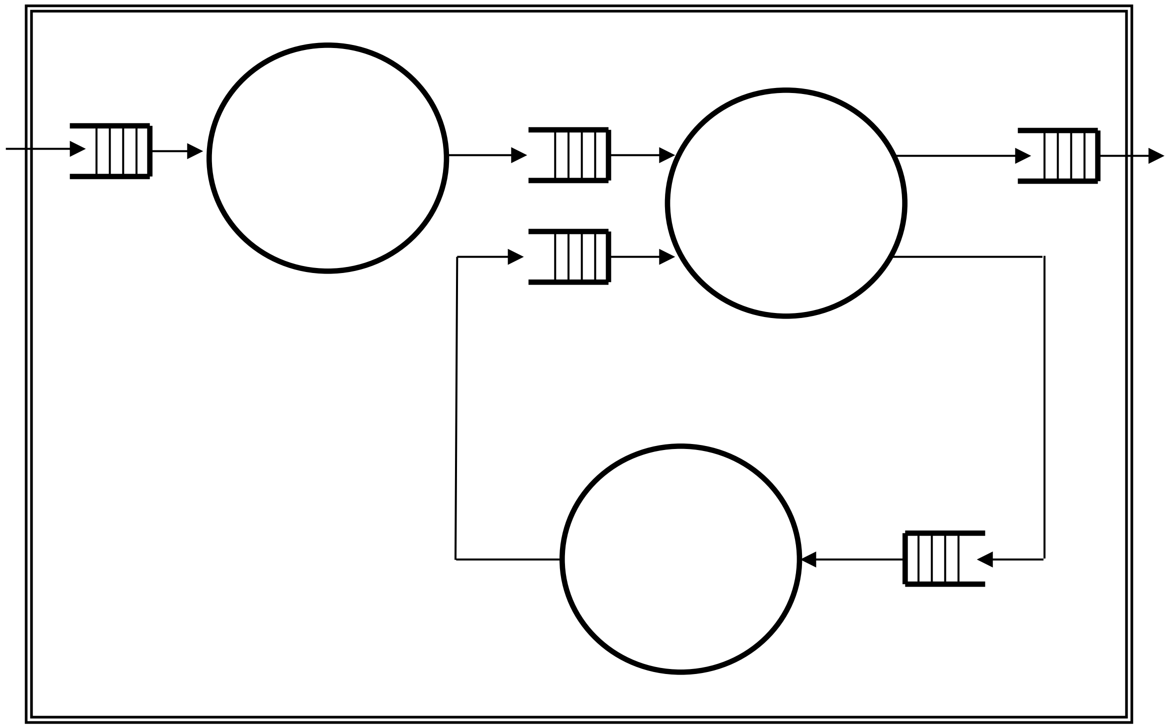
- Data flow programming with actors (Hewitt, Kahn, etc)
  - Associate resources with streams
  - Clean cut between execution specifics and algorithm design
  - Strict semantics with explicit parallelism provides foundation for analysis and model transformation
- CAL Actor Language (UC Berkeley, Xilinx) http://opendf.org
  - Part of MPEG/RVC



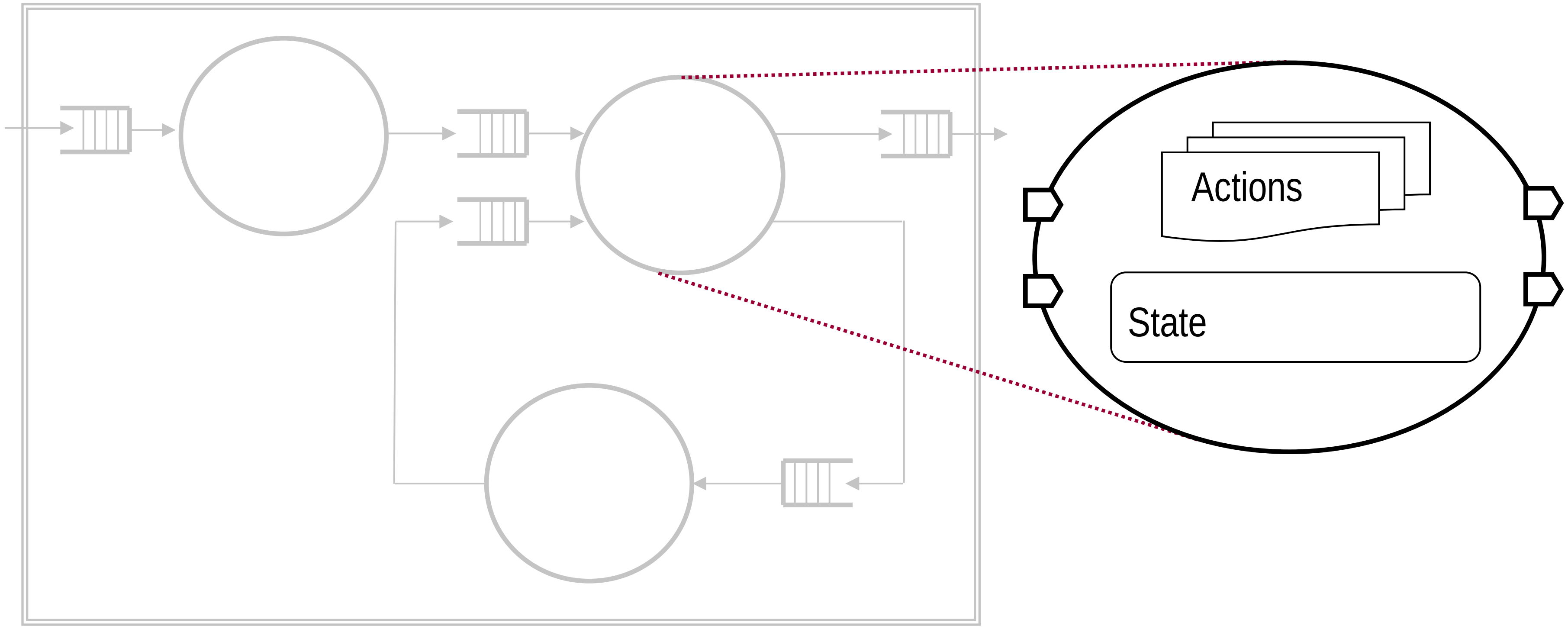
CAL (Cal Actor Language)

- A language for writing dataflow actors
  - designed at UC Berkeley in 2002/3 (Janneck & Eker)
  - compilers to hardware and software
  - standardized by MPEG/ISO in 2009
  - (subset RVC-CAL)

dataflow



actors & actions



input/output

```
actor ID () In ==> Out :

    action In: [a] ==> Out: [a] end
end
```

```
actor ID () In ==> Out :

    action [a] ==> [a] end
end
```

```
actor Add () Input1, Input2 ==> Output:

    action [a], [b] ==> [a + b] end
end
```

```
actor AddSeq () Input ==> Output:

    action [a, b] ==> [a + b] end
end
```

nondeterminism

```
actor Merge () Input1, Input2 ==> Output:

    action Input1: [x] ==> [x] end
    action Input2: [x] ==> [x] end
end
```

This actor is non-deterministic...

... and so is this.

```
actor Split () Input ==> Output1, Output2:

    action [x] ==> Output1: [x] end
    action [x] ==> Output2: [x] end
end
```

guarded actions

```
actor SplitPred (P) Input ==> Y, N:

    action [a] ==> Y: [a]
    guard P(a) end

    action [a] ==> N: [a]
    guard not P(a) end
end
```

```
actor Select () S, A, B ==> Output:

    action S: [sel], A: [v] ==> [v]
    guard sel end

    action S: [sel], B: [v] ==> [v]
    guard not sel end
end
```

actors with state

```
actor Sum () Input ==> Output:

    sum := 0;

    action [a] ==> [sum]
    do
        sum := sum + a;
    end
end
```

refers to state
at the end
of the action execution

State dependent guards & action schedules

```
actor PingPongMerge ()
      Input1, Input2 ==> Output:

    s := 0;

    action Input1: [x] ==> [x]
    guard s = 0
    do
        s := 1;
    end

    action Input2: [x] ==> [x]
    guard s = 1
    do
        s := 0;
    end
end
```

```
actor PingPongMerge ()
      Input1, Input2 ==> Output:

    A: action Input1: [x] ==> [x] end

    B: action InputB: [x] ==> [x] end

    schedule fsm s1:
        s1 (A) --> s2;
        s2 (B) --> s1;
    end
end
```

priorities (when order matters)

```
actor ProcessStream () In, Config ==> Out:

    c := initialConfig();

    action Config: [newC] ==>
    do
        c := newC;
    end

    action In: [data]
           ==> [compute(data, c)]
end
```

how to enforce firing of one action

over another?

intuition:

among the enabled actions,

one with highest priority is

fired

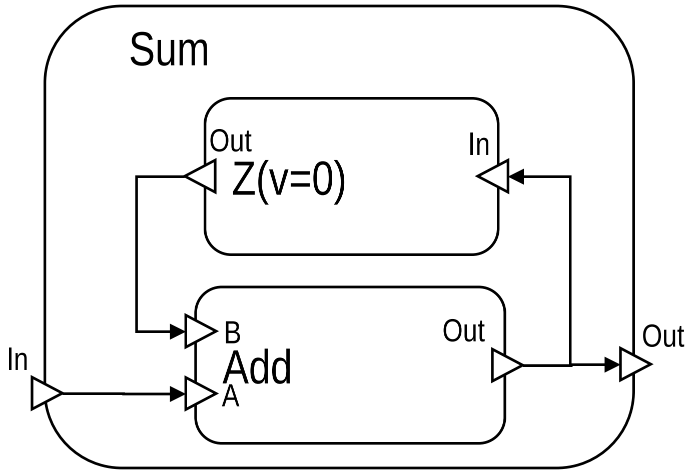
NOTE: behavior is

timing-dependent!

```
actor ProcessStream () In, Config ==> Out:

    c := initialConfig();

    config:  action Config: [newC] ==>
             do
                 c := newC;
             end

    process: action In: [data]
                    ==> [compute(data, c)] end

    priority
        config > process;
    end
end
```

building a simple network: Sum

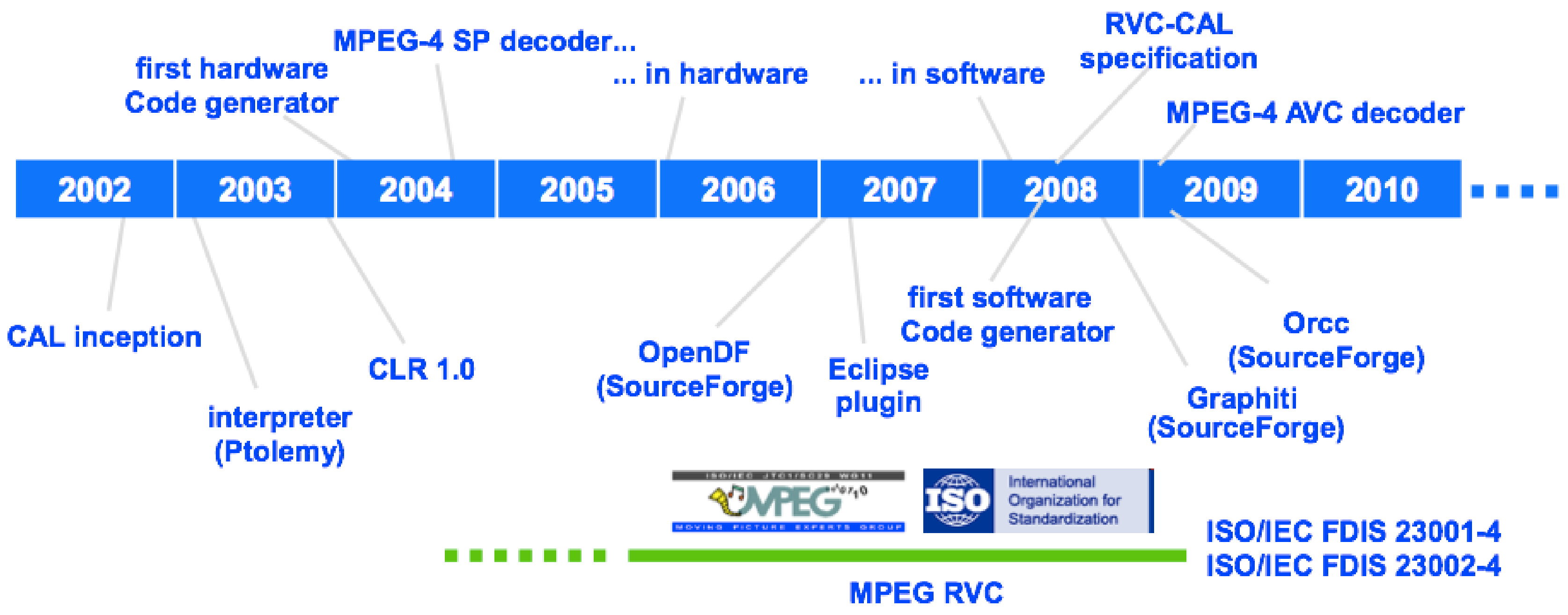
```
actor Add () A, B ==> Out:
  action [a], [b] ==> [a + b] end
end
```



```
actor Z (v) In ==> Out:

  A: action ==> [v] end
  B: action [x] ==> [x] end

  schedule fsm s0:
    s0 (A) --> s1;
    s1 (B) --> s1;
  end
end
```

```
network Sum () In ==> Out:

entities
  add = Add();
  z = Z(v=0);

structure
  In --> add.A;
  z.Out --> add.B;

  add.Out --> z.In;

  add.Out -- > Out;
end
```

CAL History



(Picture by Janneck, Jörn & Marco Mattavelli)

More Results

- CAL Actor Language
  - Standardized as part of MPEG-B RVC-CAL (ISO 23001-4)
  - Ptolemy spin-off
- Other demos
  - AMR-WB encoder, HW/SW partition support demo, 3D video, crypto library
- Documentation
  - 8+ PhD theses (EPFL, U Maryland, EPFL, Åbo)
  - 20-30 Master's Theses
  - Tons of papers.
    - http://www.hooklee.com/Research/RVC_CAL_Bibliography.html
- Groups
  - EPFL, INSA/Rennes, LTH & Halmstad, Turku/Åbo, U of Surrey
- Tools
  - OpenDF (Xilinx -> Lund University)
  - ORCC (Open RVC-CAL Compiler - Orcc)
  - Caltoopia
  - Ptolemy II

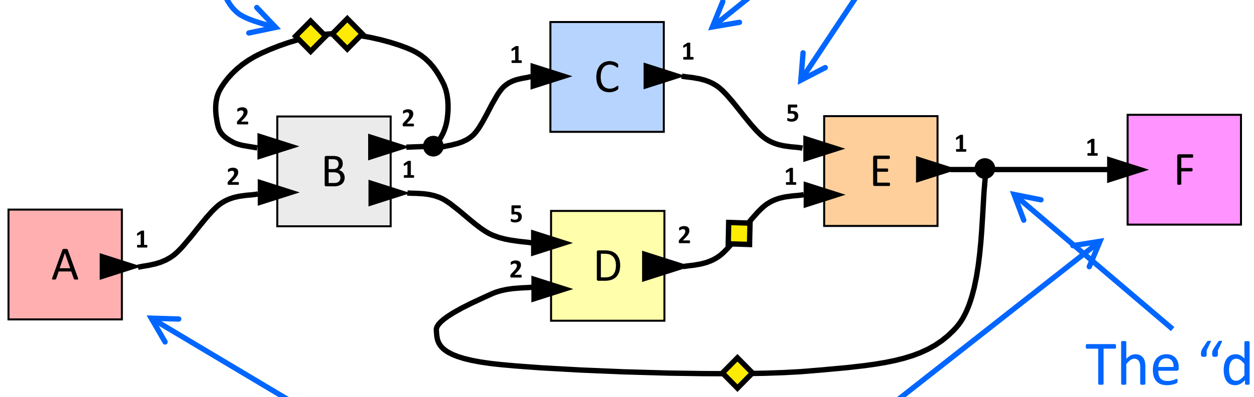
Synchronous Dataflow (SDF) [Lee87]

- Actors consume and produce a fixed number of tokens in each firing
- Allows for extensive compile-time analysis
  - Static scheduling (no risk of deadlock)
  - Static allocation of buffers (no unbounded buffering)
  - Possible to reason about performance metrics
- Possible to generate tight code

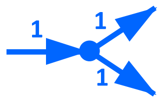
SDF Example

There may be cycles,
but initial tokens (delays)
are required to avoid deadlock

Token rates are shown
at input/output ports

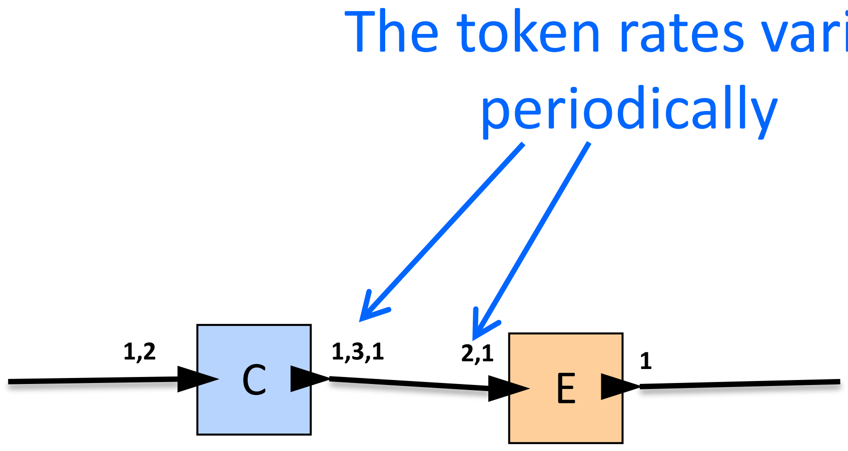


The “dots” are
duplicators



There may be sources and
sinks

Cyclo-Static Dataflow (CSDF)



Also statically schedulable

Dynamic dataflow (DDF) [Lee95]

- Data dependent action firing
- Not statically schedulable

ACTORS: Model Transformations

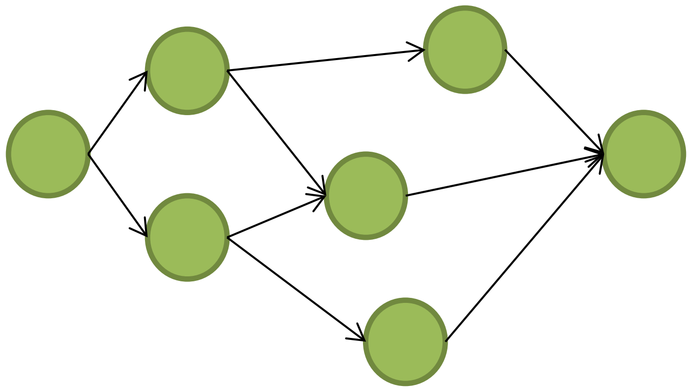
- Merging of actors within statically schedulable regions
  - Faster execution on single processors
- Splitting of actors
  - Express fine-grained parallelism
  - FPGA platforms

ACTORS Applications

- Eventually, the ACTORS resource management approach should be applicable to arbitrary applications
- Initially, only CAL applications
- Two types:
  - Static CAL applications
  - Dynamic CAL applications

Static CAL Applications

- Dataflow graph can be mapped to a static precedence graph (DAG)
  - Task activation graph
- "SDF-like" dataflow graphs
- E.g. control applications
- Schedulability theory can be applied
  - Assuming WCET estimates
  - Generates reservation parameters



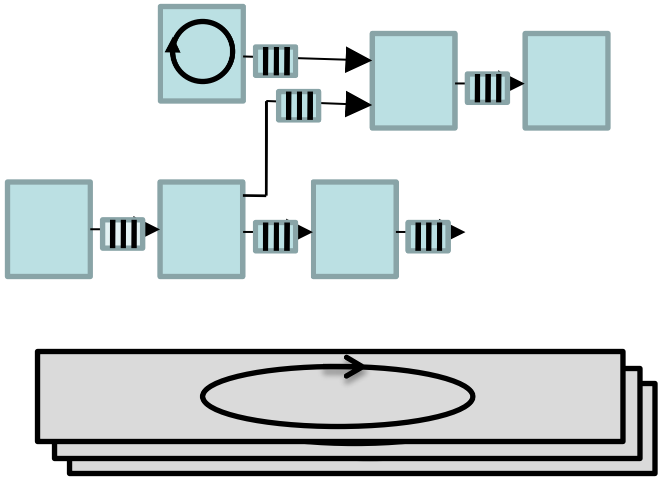
Dynamic CAL Applications

- Dataflow graph cannot be mapped to a static precedence graph
- Best-effort scheduling
- E.g. multimedia applications

CAL Run-Time System

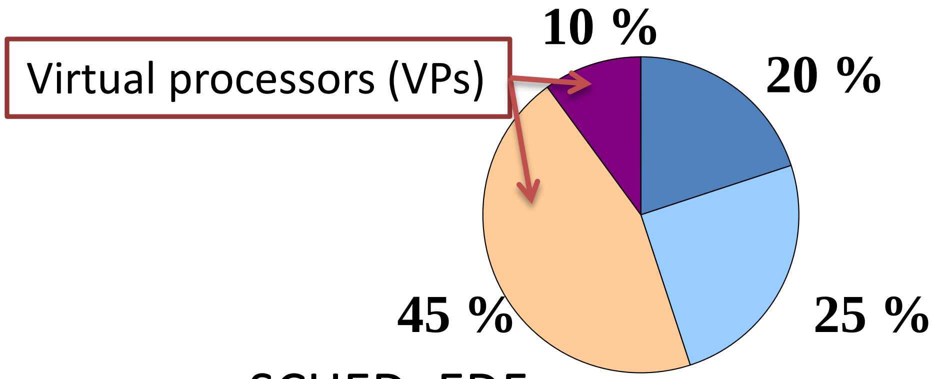
- The execution environment for CAL applications
- Provides system actors for interfacing to environment
- Dynamic CAL applications
  - Run-time thread
    - While (1) {

```
Select actor();
Execute actor();
}
```

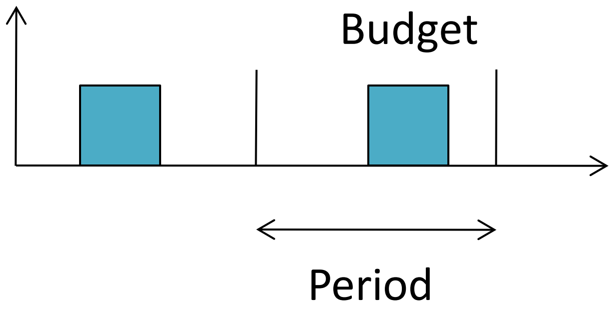


ACTORS: Key Ingredients

3. Reservation-Based CPU Scheduling



- Periodic Bandwidth Servers
  - Constant Bandwidth Server



   – SCHED_EDF
     - Partitioned multi-core EDF scheduler
     - Hard CBS reservations
     - Each reservation may contain several threads
     - Hierarchical scheduler with SCHED_EDF tasks executing on a higher level than ordinary Linux tasks

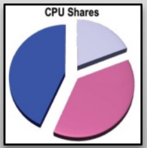
4. Multicore Linux Platforms
   – ARM 11, x86

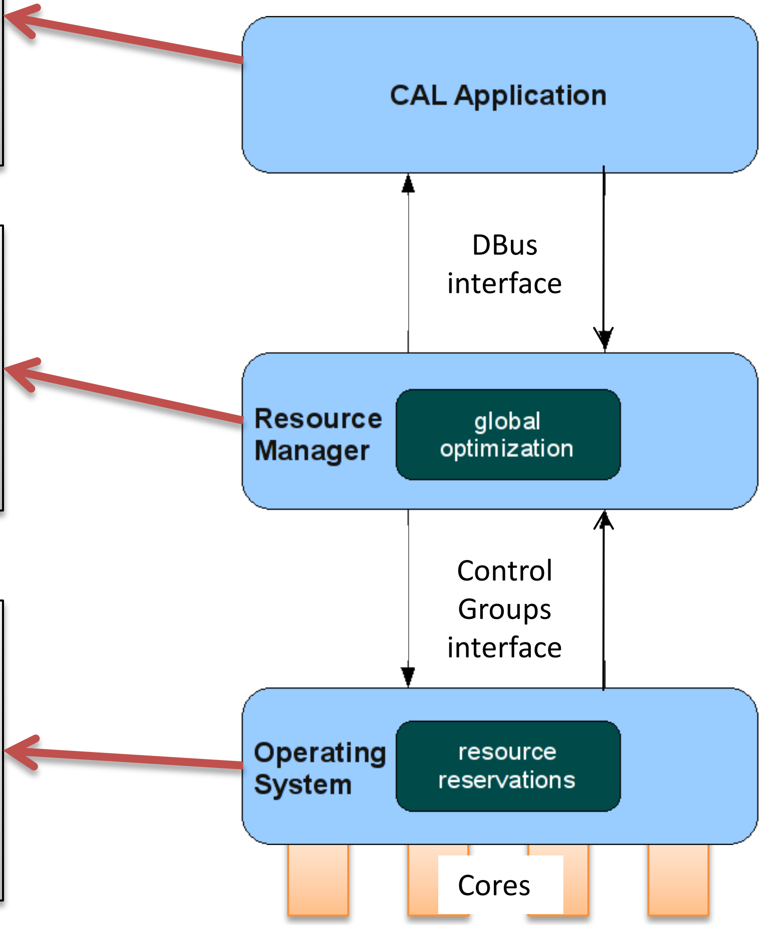
Overview

- CAL Dataflow Applications
    - Dataflow run-time system
- Legacy applications through wrapper
    - All threads within one VP

Resource Manager
- C++ framework
- DBus IPC to application
- Control groups API to scheduler

SCHED_EDF hierarchical scheduler
- Partitioned multi-core scheduler
- Hard CBS Reservations
  - one or several threads





Static Information

From applications to RM at registration:

- Service Level Table

Service Level	QoS	BW Requirement	BW distribution	Timing Granularity
0	100	240	60-60-60-60	20 ms
1	75	180	45-45-45-45	20 ms
2	40	120	30-30-30-30	20 ms

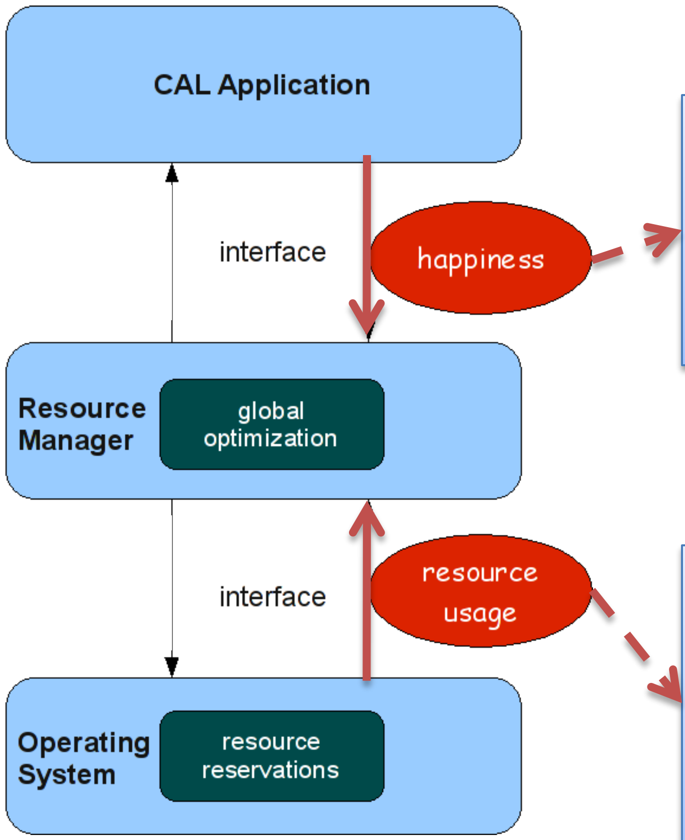
- Thread IDs and how they should be grouped

From system administrator to RM at startup:

- Application Importance

Appl.	Importance
Appl 1	10
Appl 2	20
Appl 3	100
Default	10

Dynamic Inputs



Happiness:

- boolean indicator of whether the QoS obtained corresponds to what could be expected at the current service level

Used Budget (Bandwidth):

- average used budget

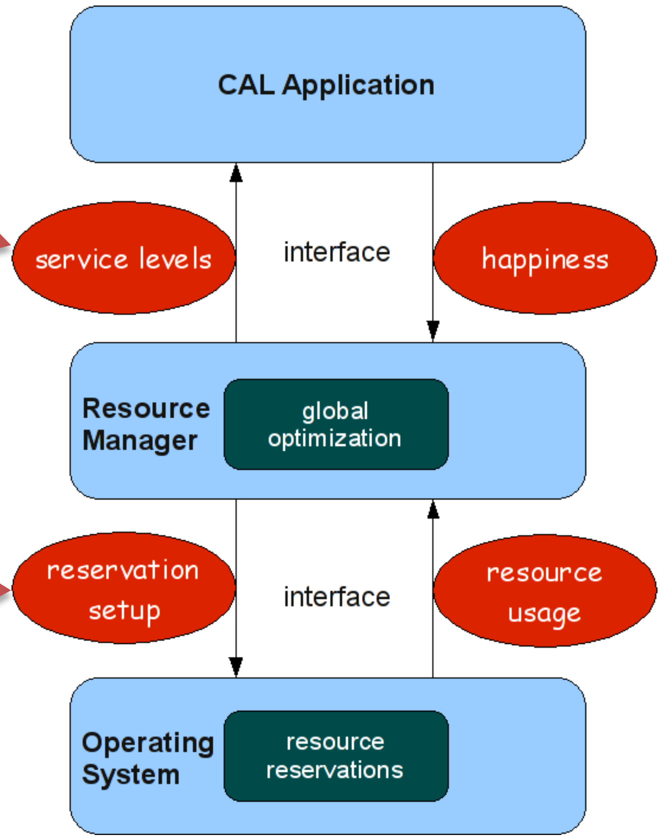
Exhaustion Percentage:

- percentage of server periods in which the budget was exhausted

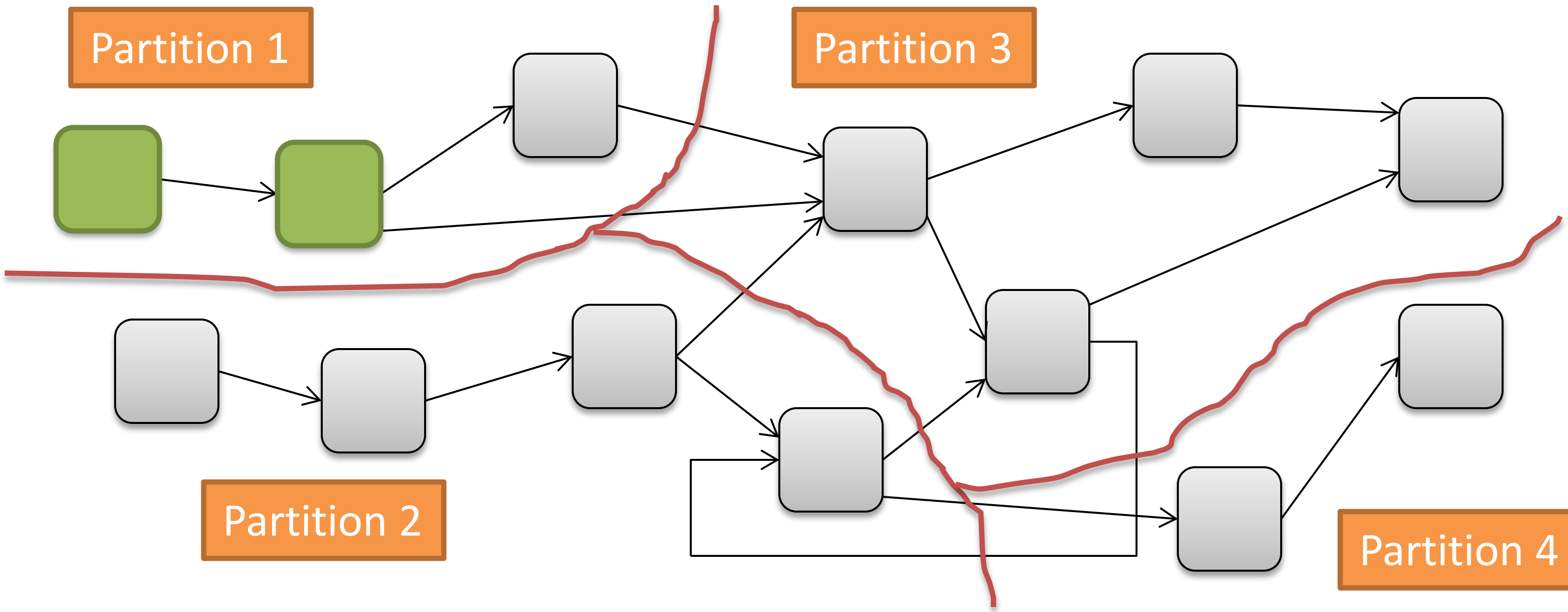
Outputs

Service Level

Reservation Parameters:
- Budget
- Period
- Affinity
  - RM may migrate VPs

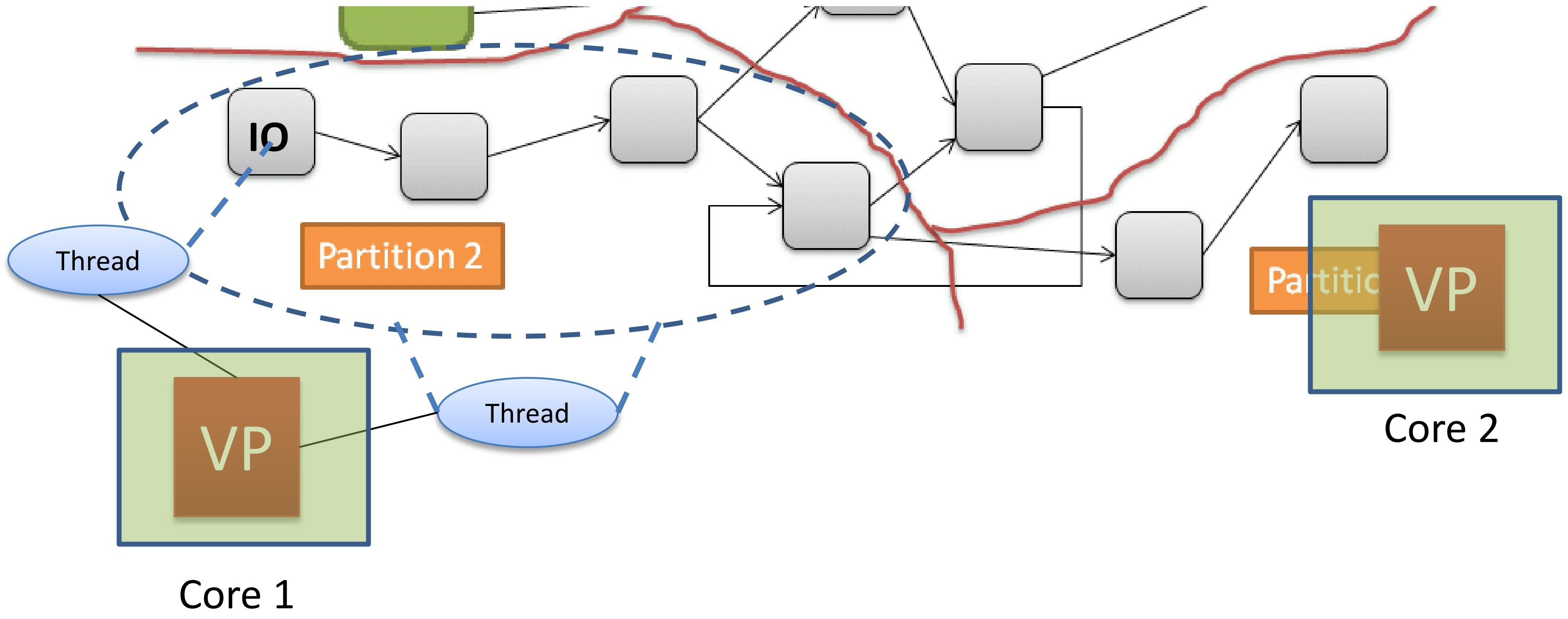


Dataflow Analysis



- Static partitioning
- Automatic analysis for SDF/CSDF actors
- Automatic merging of SDF/CSDF actors to improve run-time performance

CAL Run-Time System



- One thread per partition executing its actors using round-robin
- One thread per "system actor" (IO, time)
- The threads from the same partition are executed by the same virtual processor
- If possible the VPs are mapped to different physical cores in order to enable parallel execution

Resource Manager Functionality

- Assign service levels
  - When applications register or unregister
  - Formulated as a ILP problem
    - Importance as weight
  - glpk solver

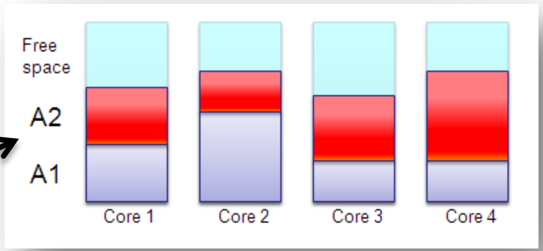
$$\max \sum_{i=1}^{n} w_i q_i x_i$$

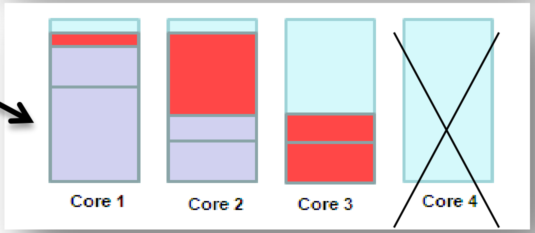
$$\sum_{i=1}^{n} \alpha_i x_i \leq C$$

$$\forall i, \sum x_i = 1$$

- Mapping & bandwidth distribution
  - Map reservations to cores
  - Distribute the total BW to the reservations

    Two Approaches:
    - Spread out the VPs and balance the load
    - Pack the VPs in as few cores as possible
      - Allow turning off unused cores
  - Bin packing





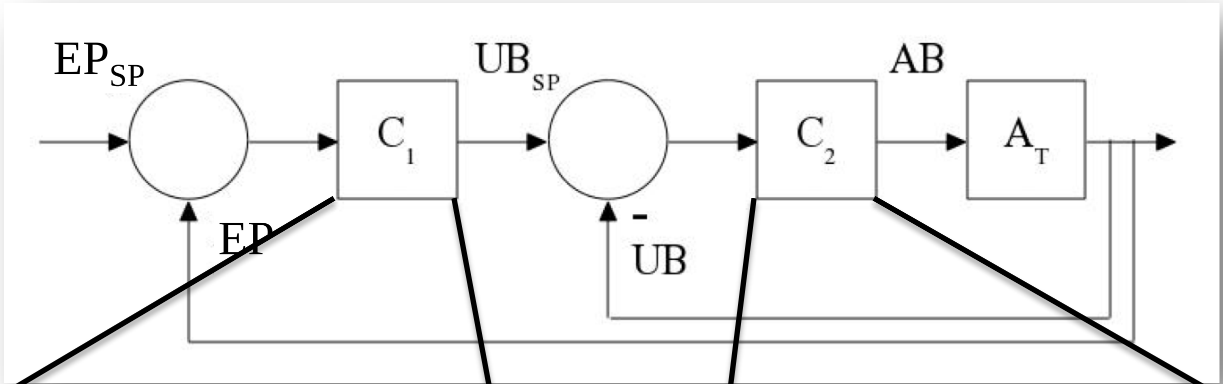
- At most 90% of each core is used for SCHED_EDF tasks

Resource Manager Functionality

- Separate service level assignment and BW Distribution
  - The best service level assignment may lead to an unfeasible BW distribution
  - Approach 1:
    - New SL assignment that generates the next best solution
    - New BW Distribution
  - Approach 2:
    - Compress the individual VPs

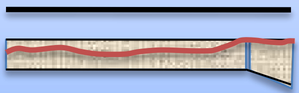
Resource Manager Tasks

- Bandwidth adaptation
  - Adjust the servers bandwidth dynamically based on measured resource usage and obtained happiness

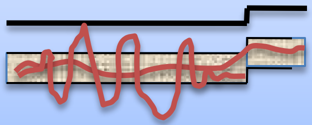


  - bandwidth is happy

Changes what is meant by sufficiently close based on EP:

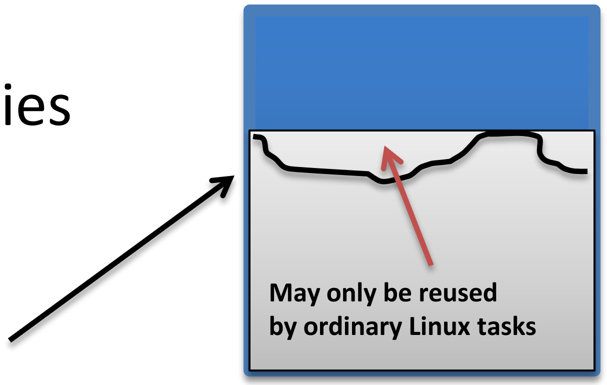


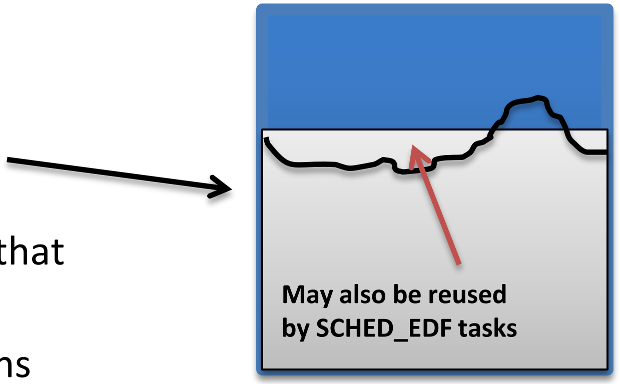
Changes the AB so that the UB lies sufficiently close:



Resource Manager Tasks

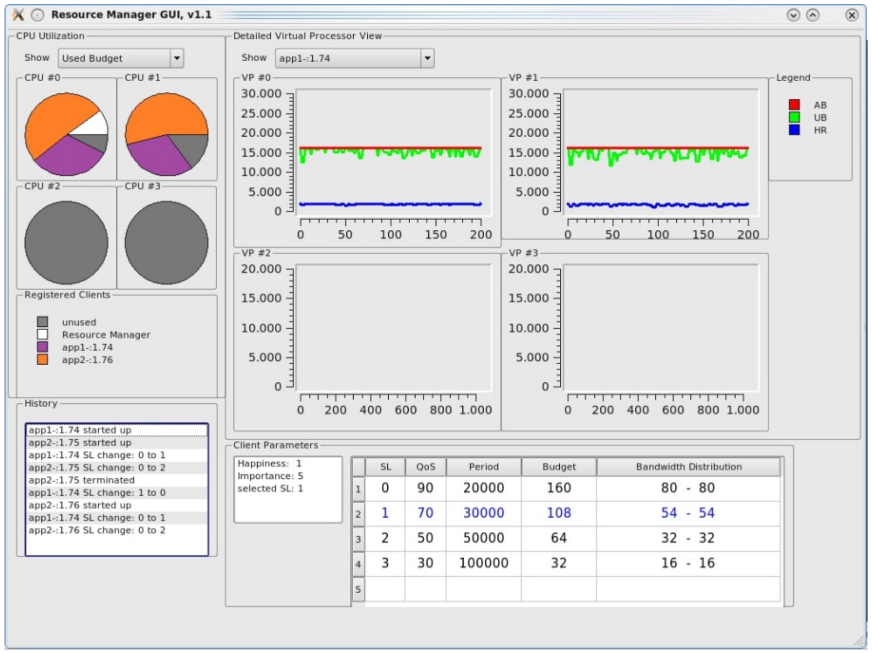
- Multiple bandwidth adaptation strategies
- Strategy 1:
  - A VP may never consume more bandwidth than what was originally assigned to it
  - BW controller may reduce the BW if not used
- Strategy 2:
  - A VP may use more resources than originally assigned to it
    - If there are free resources available, or
    - If there are VPs of less important applications that use more BW than originally allocated to them
    - In the latter case the less important applications are compressed
    - All applications are always guaranteed to obtain their originally assigned values (can never be compressed beyond that)





RM Support Software

- GUI
  - VP to core assignment
  - AB, UB, and EP
  - Service Level Table
  - Event history
  - Itself an application under the control of the RM
- Load Generator
  - Generates artificial load for testing
- Application Wrapper
  - Wrapper for non-Actors aware applications



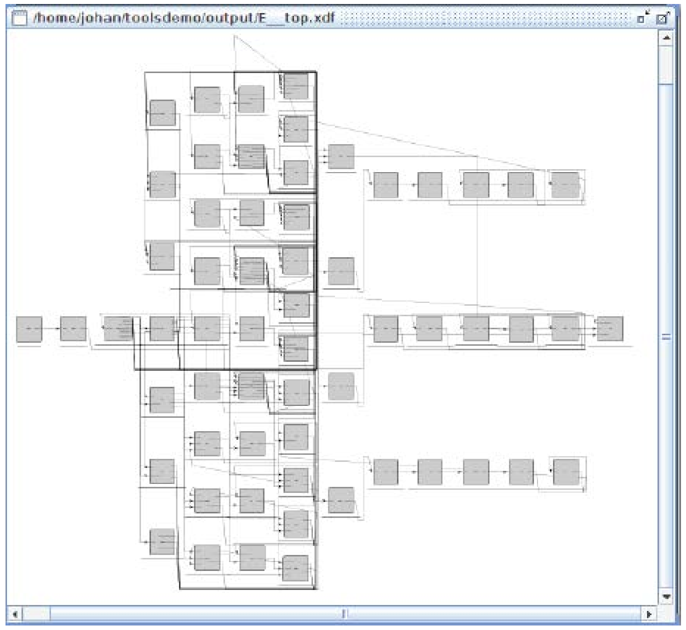
```
app1-:1.74 started up
app2-:1.75 started up
app1-:1.74 SL change: 0 to 1
app2-:1.75 SL change: 0 to 2
app2-:1.75 terminated
app1-:1.74 SL change: 1 to 0
app2-:1.76 started up
app1-:1.74 SL change: 0 to 1
app2-:1.76 SL change: 0 to 2
```

Happiness: 1
Importance: 5
selected SL: 1

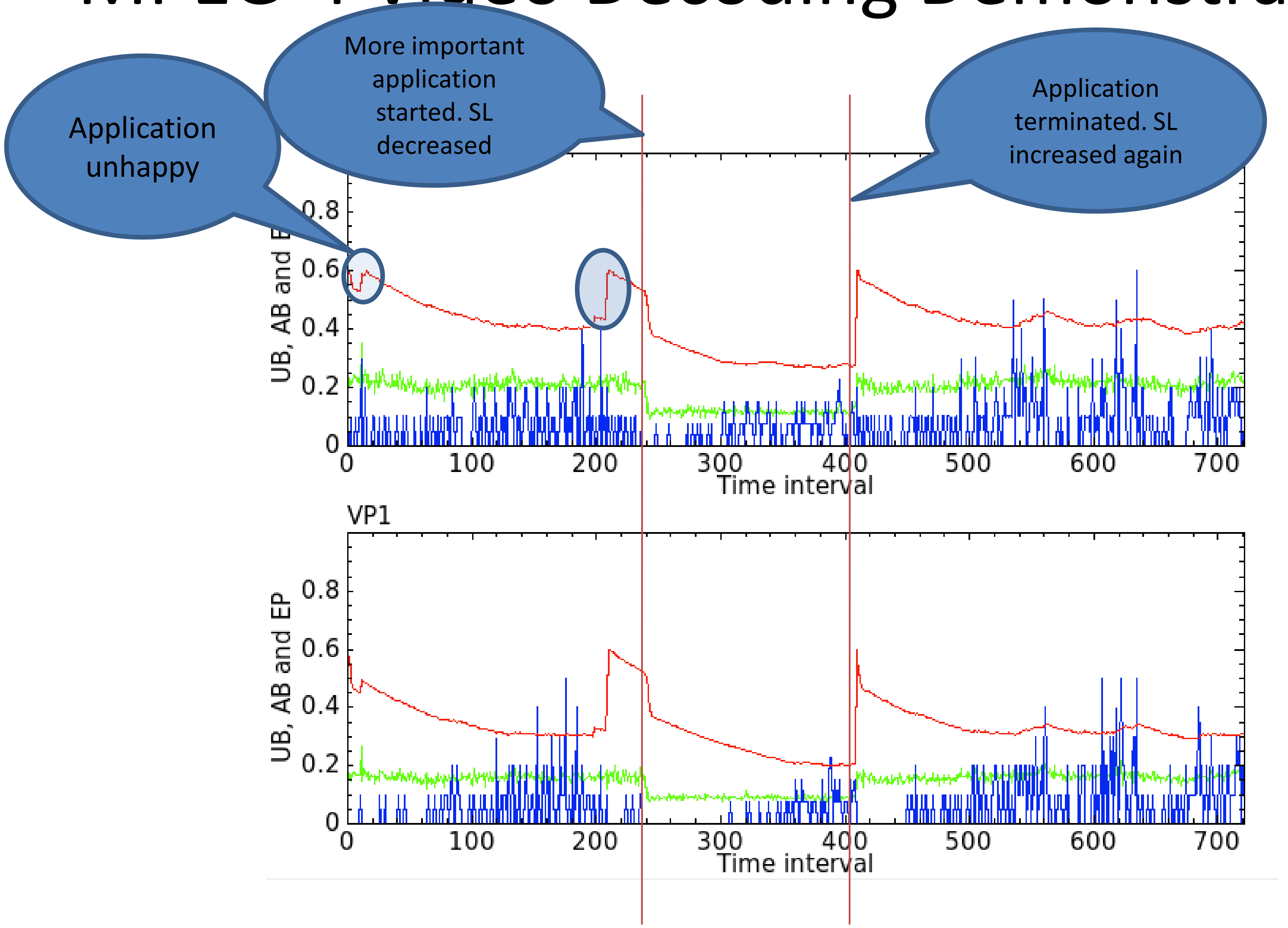
	SL	QoS	Period	Budget	Bandwidth Distribution
1	0	90	20000	160	80 - 80
2	1	70	30000	108	54 - 54
3	2	50	50000	64	32 - 32
4	3	30	100000	32	16 - 16
5					

MPEG-4 Video Decoding Example

- MPEG-4 SP decoder implemented in CAL
- Connected to an Axis network camera
- Service level changes results in commands from the decoder to the camera to reduce the frame rate and/or resolution

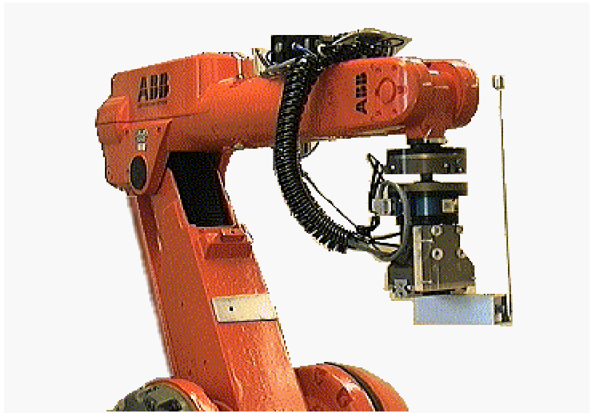


MPEG-4 Video Decoding Demonstrator

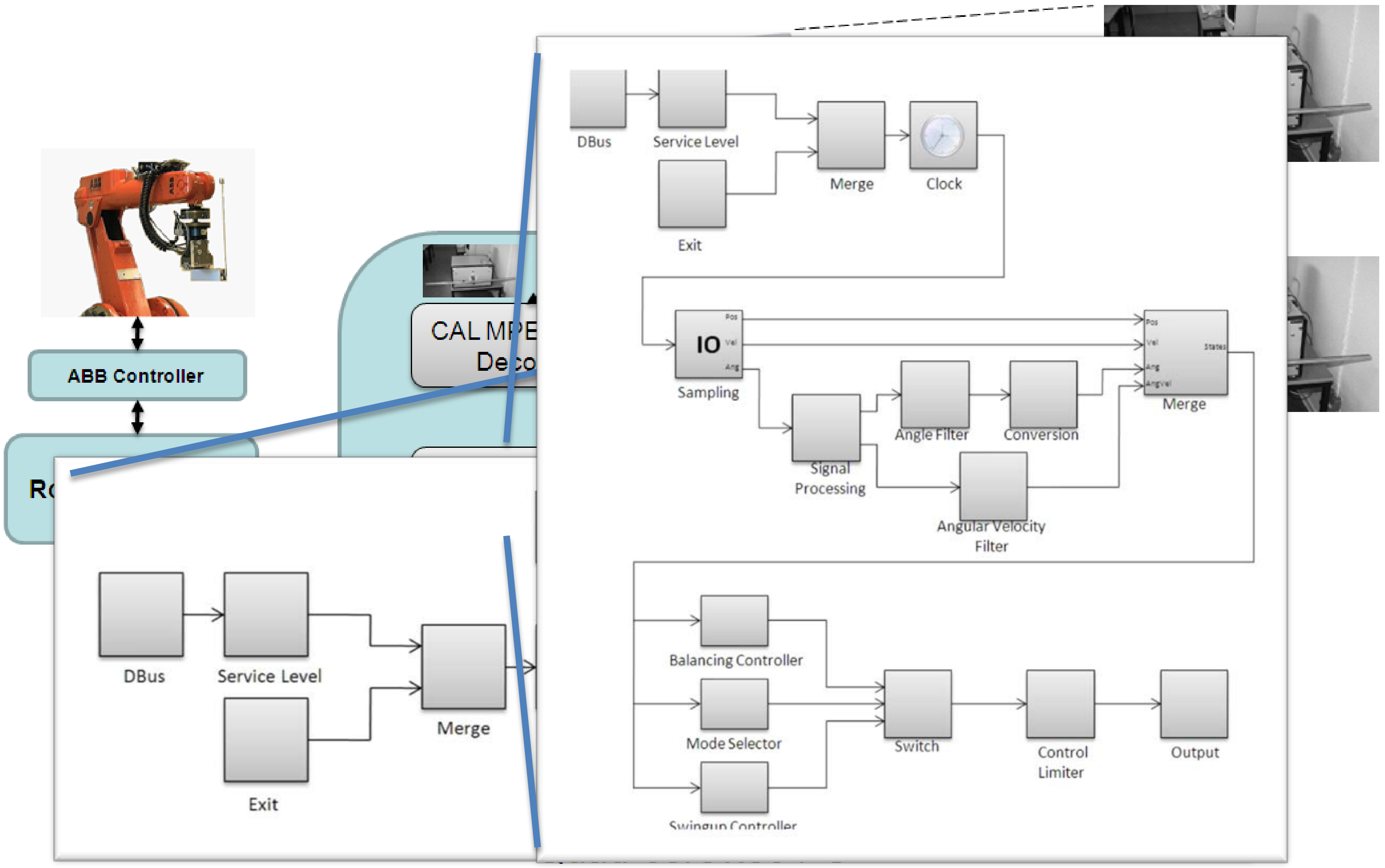


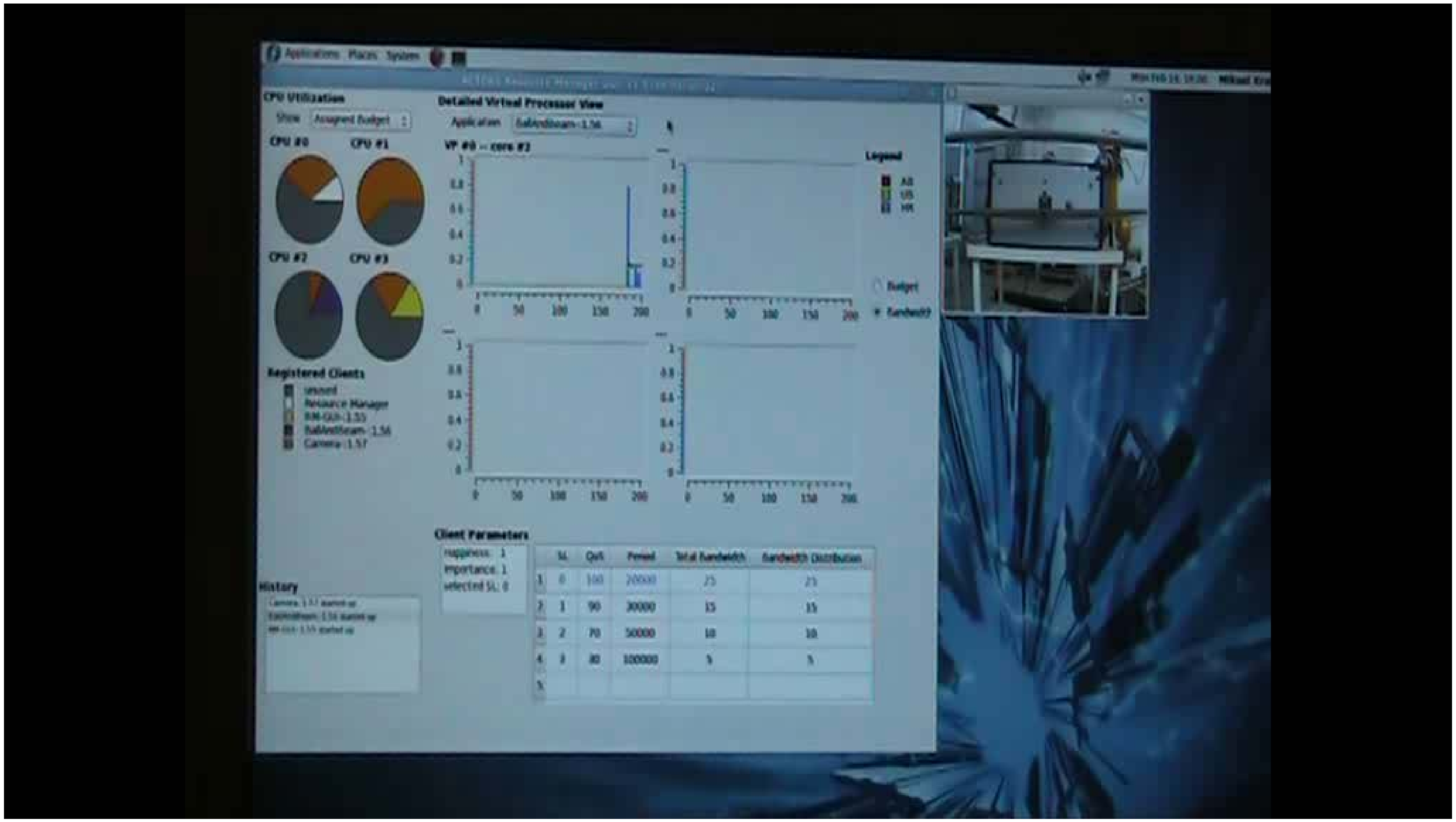
Feedback Control Demonstrator

- Industrial robot balancing an inverted pendulum
  - Pendulum controller in CAL
  - Service level changes correspond to changes in sampling period
- Ball and Beam Processes
  - Controller in CAL
  - Service level changes correspond to changes in sampling period
- MPEG-4 Video Decoder
  - Service level changes correspond to changes in resolution/frame rate



Feedback Control Demonstrator





Client Parameters

Happiness: 1
Importance: 1
selected SL: 0

	SL	QoS	Period	Total Bandwidth	Bandwidth Distribution
1	0	100	20000	25	25
2	1	90	30000	15	15
3	2	70	50000	10	10
4	3	40	100000	5	5
5					

Drawbacks

- ILP does not scale well
- Requires a lot of information from the applications
- More natural if the applications select their service levels and the resource manager adjust the vp size

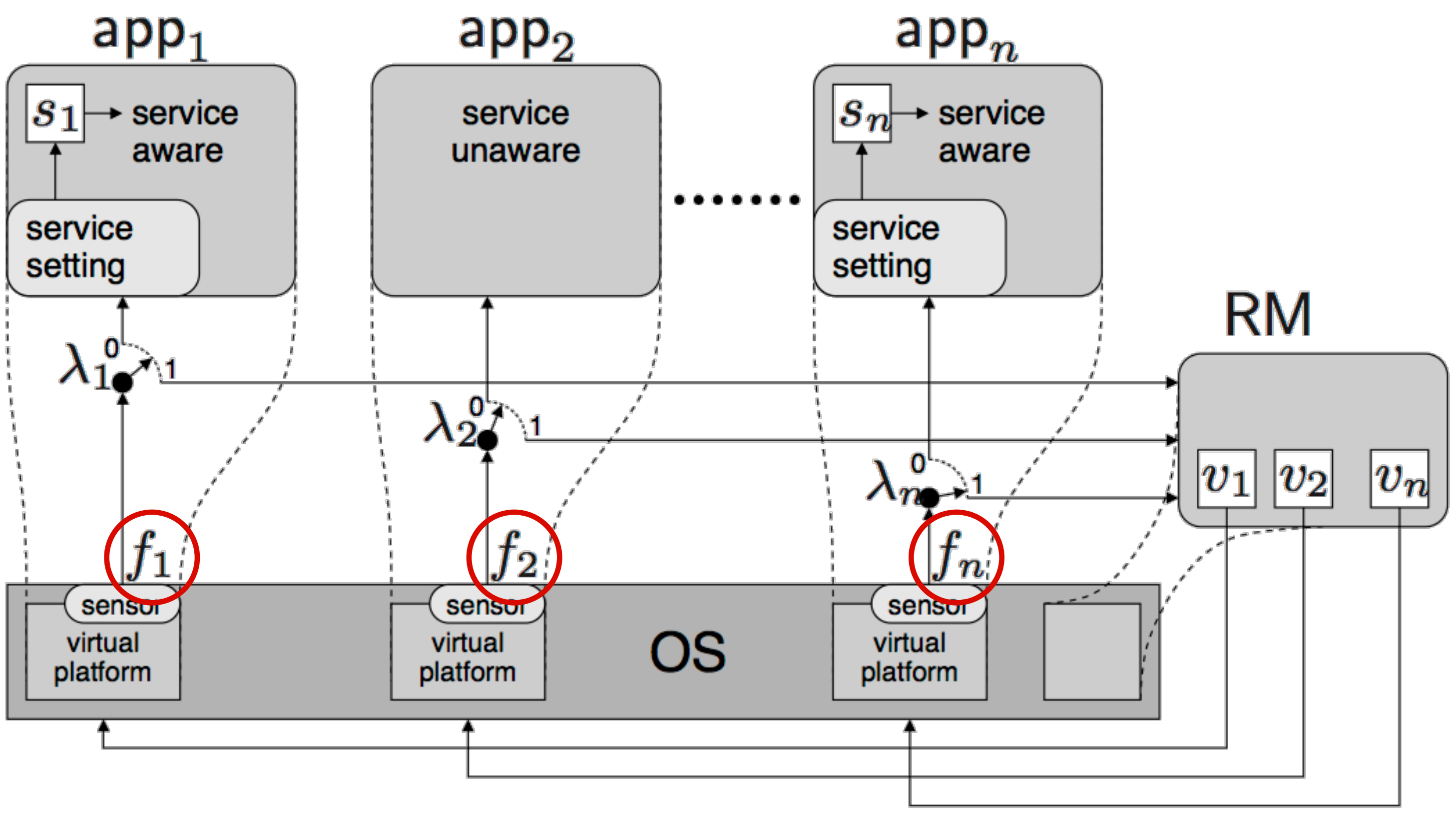
→ The game-theory inspired approach

Content

- Cyber-Physical Systems – My Personal View
- Control of Computer Systems
  - Motivation and Background
  - A simple queue length control example
- **Resource Management for Multi-Core Embedded Systems**
  - The ACTORS Resource Manager
    - Video demo
  - **Game-Theory Resource Manager**
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications
- A CPS Story

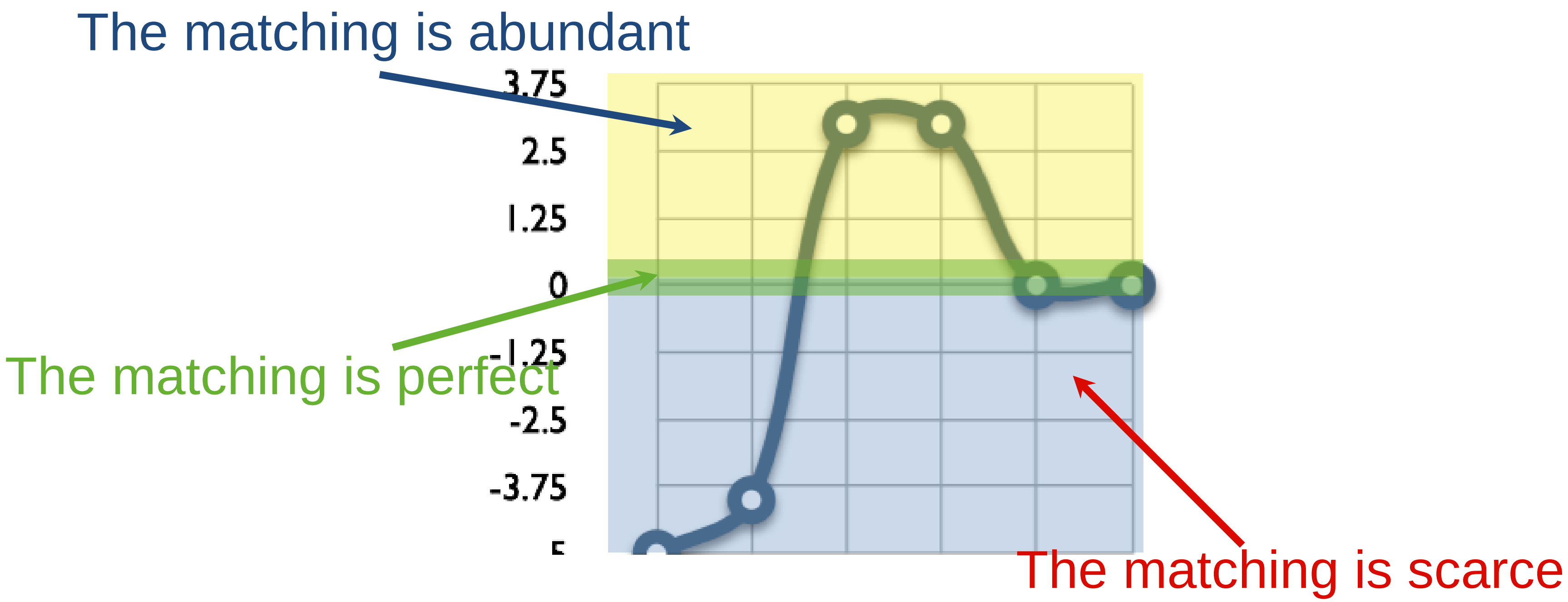
Towards decentralization

- The resource manager allocates resources and applications choose their service levels based on current performance

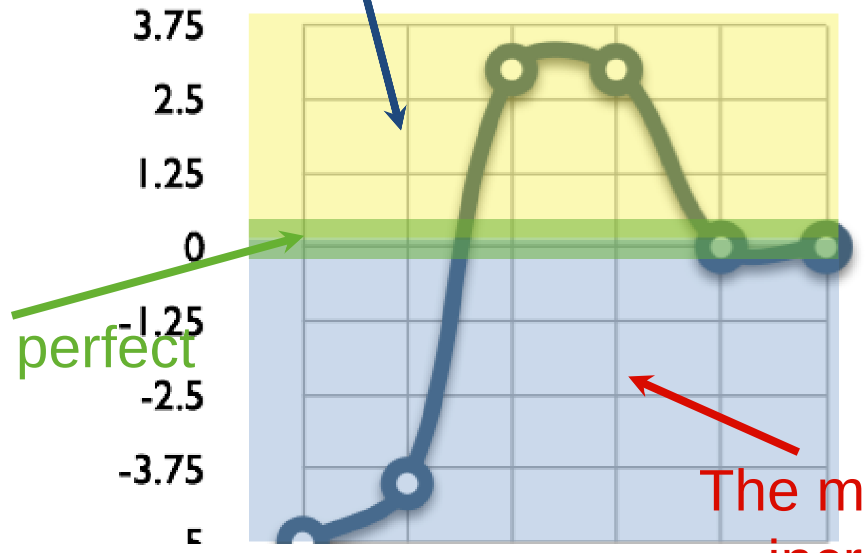


Matching function

Measures how well the resources assigned to the application (vp size) matches the resource requirements (SL) of the applications



The matching is abundant
- decrease vp or
- increase SL



The matching is perfect

The matching is scarce
- increase vp or
- decrease SL

Application weights

- $\lambda_i \in [0..1]$
- $\lambda_i = 0$ means that only the application should adjust
- $\lambda_i = 1$ means that only the resource manager should adjust

Matching Function

- Application executes a series of jobs
  - Execution time $C_i = a_i s_i$
  - Response time for each job $R_i = \frac{C_i}{v_i} = \frac{a_i s_i}{v_i}$
  - Want this to be equal to the deadline D_i
  - Hence

$$f_i = \frac{D_i}{R_i} - 1 = \frac{D_i v_i}{a_i s_i} - 1 = \beta_i \frac{v_i}{s_i} - 1$$

  - Can be measured from job start and stop times
  - Depends on service level and virtual processor speed

Resource Manager

At each step:

- Measure f_i for all the applications
  - The applications report the start and stop of each job by writing to shared memory
  - RM reads from shared memory and calculates the response time
- Updates the virtual processors:

$$v_i(k+1) = v_i + \varepsilon_{RM}(k)\left(-\lambda_i f_i(k) + \sum_{j=1}^{n} \lambda_j f_j(k) v_i(k)\right)$$

$$\varepsilon_{RM}(k) = \frac{1}{k+1}$$

Service Level Adjustment

- Should set s_i so that f_i becomes 0
- Naive approach: $s_i(k+1) = \beta_i v_i(k+1)$
  - Assumes knowledge of β_i
- Instead estimate β_i

$$\beta_i = (1 + f_i(k))\frac{s_i(k)}{v_i(k)}$$

- Gives

$$s_i(k+1) = (1 + f_i(k))\frac{v_i(k+1)}{v_i(k)}s_i(k)$$

  - Continuous service levels
  - Robustified

Game Theory

- Provides convergence results and results about stationary values
- For example
  - A stationary point satisfies the following condition
    - The matching function is zero

    OR

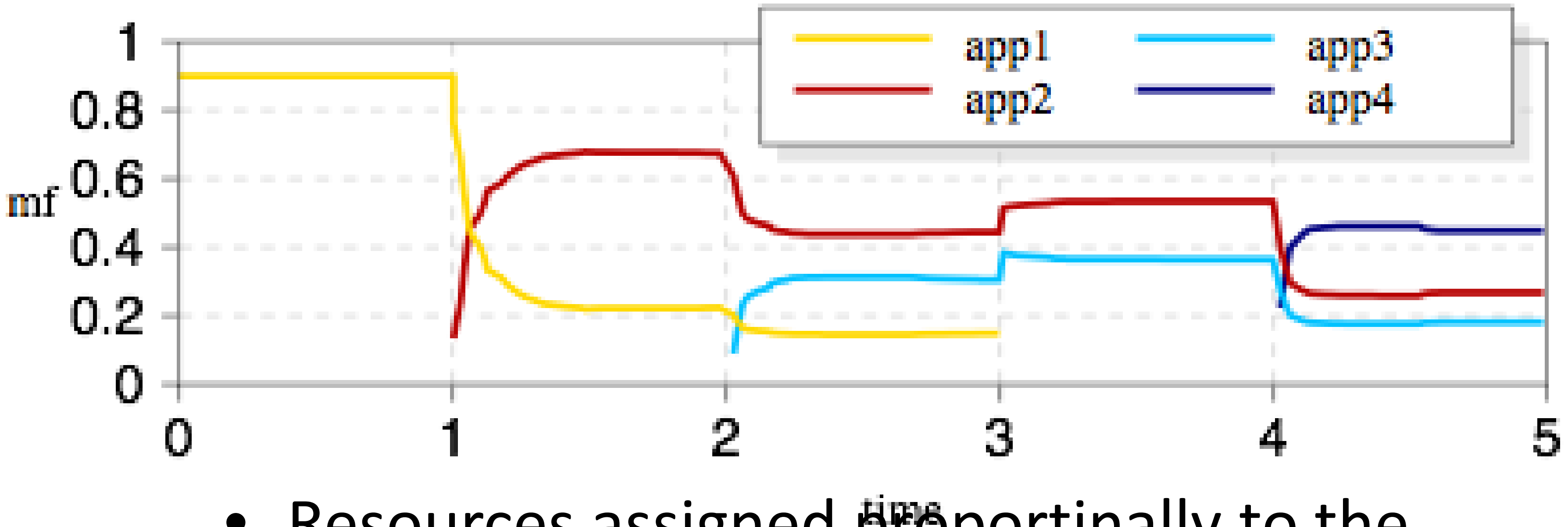
    - The service level is the smallest possible AND the matching function is negative

Implementation

- Using SCHED_DEADLINE
  - New EDF+CBS scheduling policy in Linux
  - Developed by Evidence within ACTORS

Evaluation

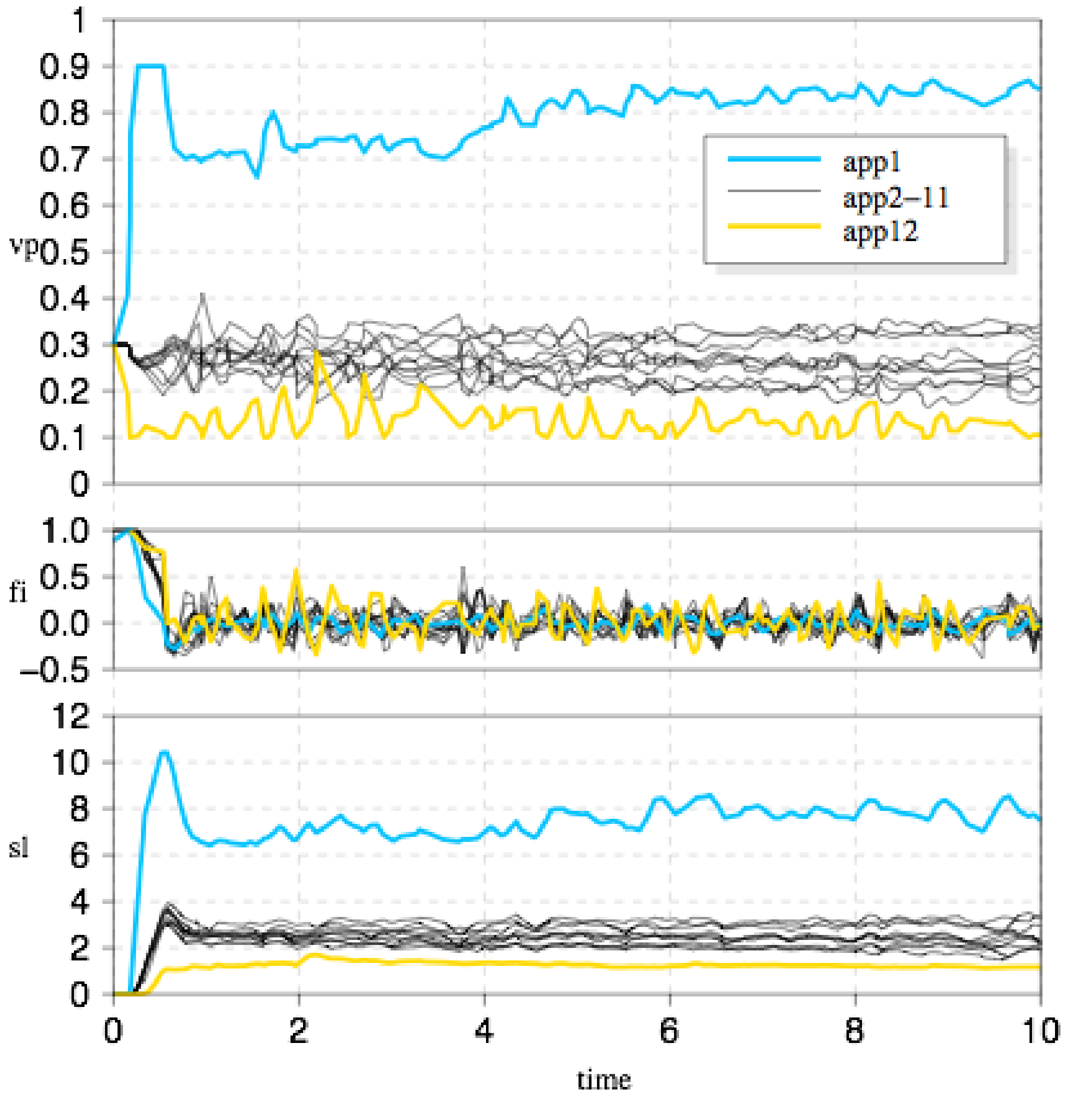
- Convergence of virtual processors
- Four applications
  λ1 = 0.1, λ2 = 0.3, λ3 = 0.2 and λ4 = 0.5



- Resources assigned proportinally to the normalized weights

Evaluation

- Multicore
- 12 identical apps
- Different weights
  - 1 – high
  - 12 – low
  - 2-11 – medium
- Resources proportional to the service levels



Content

- Cyber-Physical Systems – My Personal View
- Control of Computer Systems
  - Motivation and Background
  - A simple queue length control example
- Resource Management for Multi-Core Embedded Systems
  - The ACTORS Resource Manager
    - Video demo
  - Game-Theory Resource Manager
- **The Cloud**
  - **Problems and Challenges**
  - **Brownout-inspired resource management for web-service applications**
- A CPS Story

- As soon as you use any networked computing unit (laptop, smart phone, sensor device, …) the likelihood that the computations will be performed in a data center somewhere in the cloud, rather than locally, is very large
- News, mail, photos, tickets, books, clothes, maps, social media, television, music, banking, administrative systems, technical software, …….

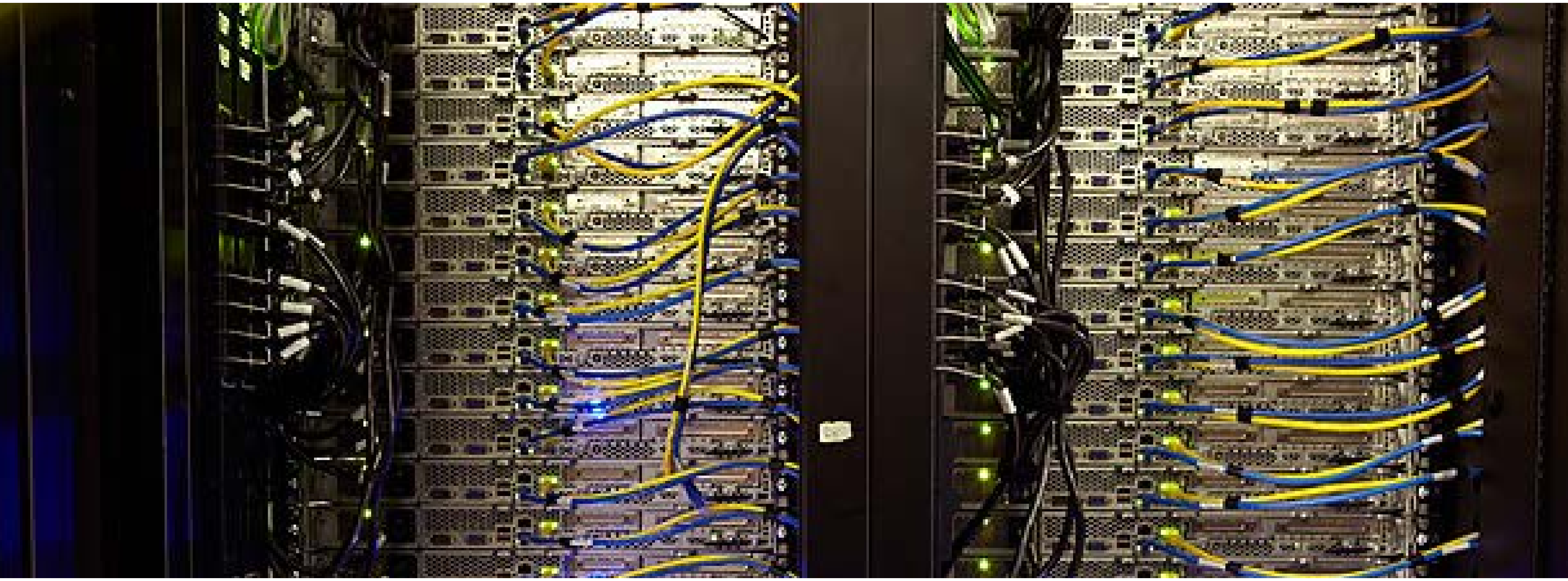
The Datacenter

What's inside?

Rack or blade-mounted servers

A modern Amazon data center consists of 50-80.000 physical servers
Each server typically has 4-8 cores

What's inside?


Networking

What's inside?

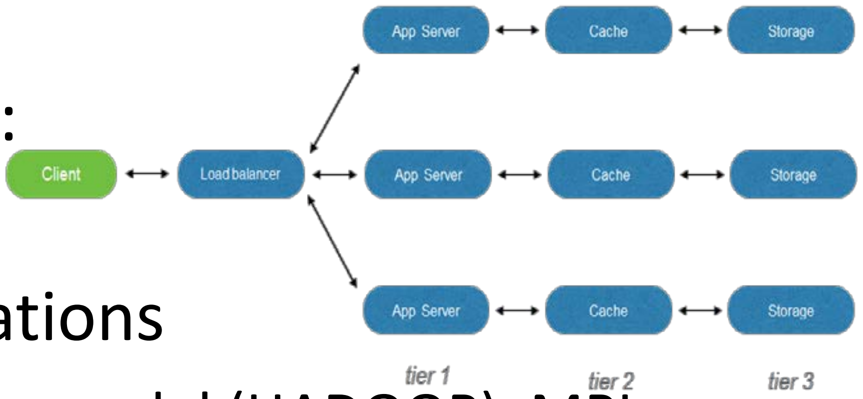
Power supplies

What's inside?

Cooling

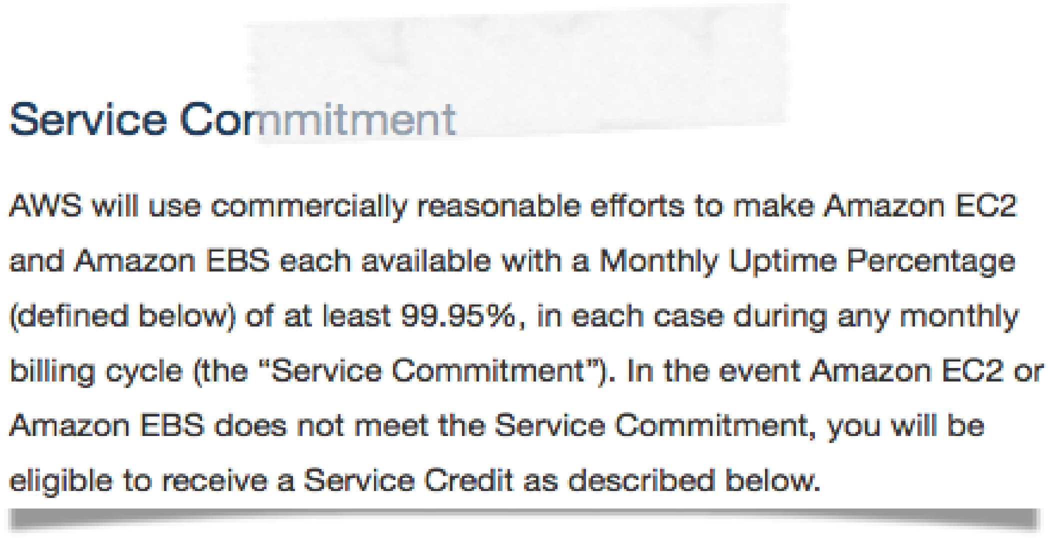
What runs in the cloud?

- Everything that can execute on physical hardware can in principle do so on virtual hardware in the cloud
  - But constraints on latency, throughput, IO, …
- Two major classes:
  - Web service applications:
    - E-Commerce
  - Massively parallel applications
    - Map/Reduce programming model (HADOOP), MPI
    - Google, Facebook, Twitter, …..



Problems with the Cloud

- Low level of determinism and predictability
  - No hard performance guarantees



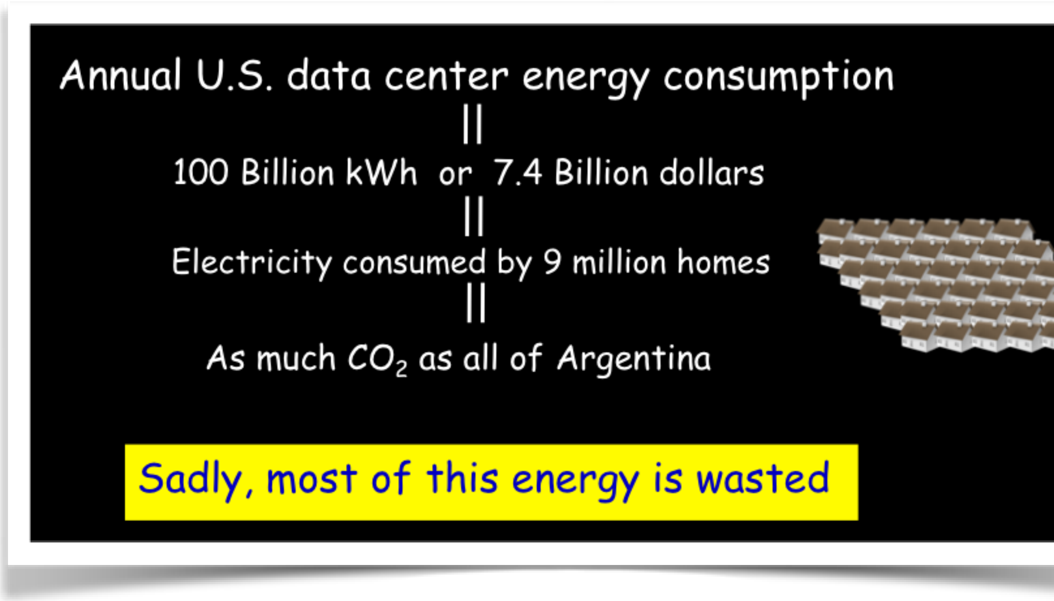
Service Commitment

AWS will use commercially reasonable efforts to make Amazon EC2 and Amazon EBS each available with a Monthly Uptime Percentage (defined below) of at least 99.95%, in each case during any monthly billing cycle (the "Service Commitment"). In the event Amazon EC2 or Amazon EBS does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below.

Problems with the Cloud

- Low level of determinism and predictability
  – No hard performance guarantees
- Data centers consume a lot of energy

Cloud Energy Consumption



Annual U.S. data center energy consumption

||

100 Billion kWh or 7.4 Billion dollars

||

Electricity consumed by 9 million homes

||

As much CO_2 as all of Argentina

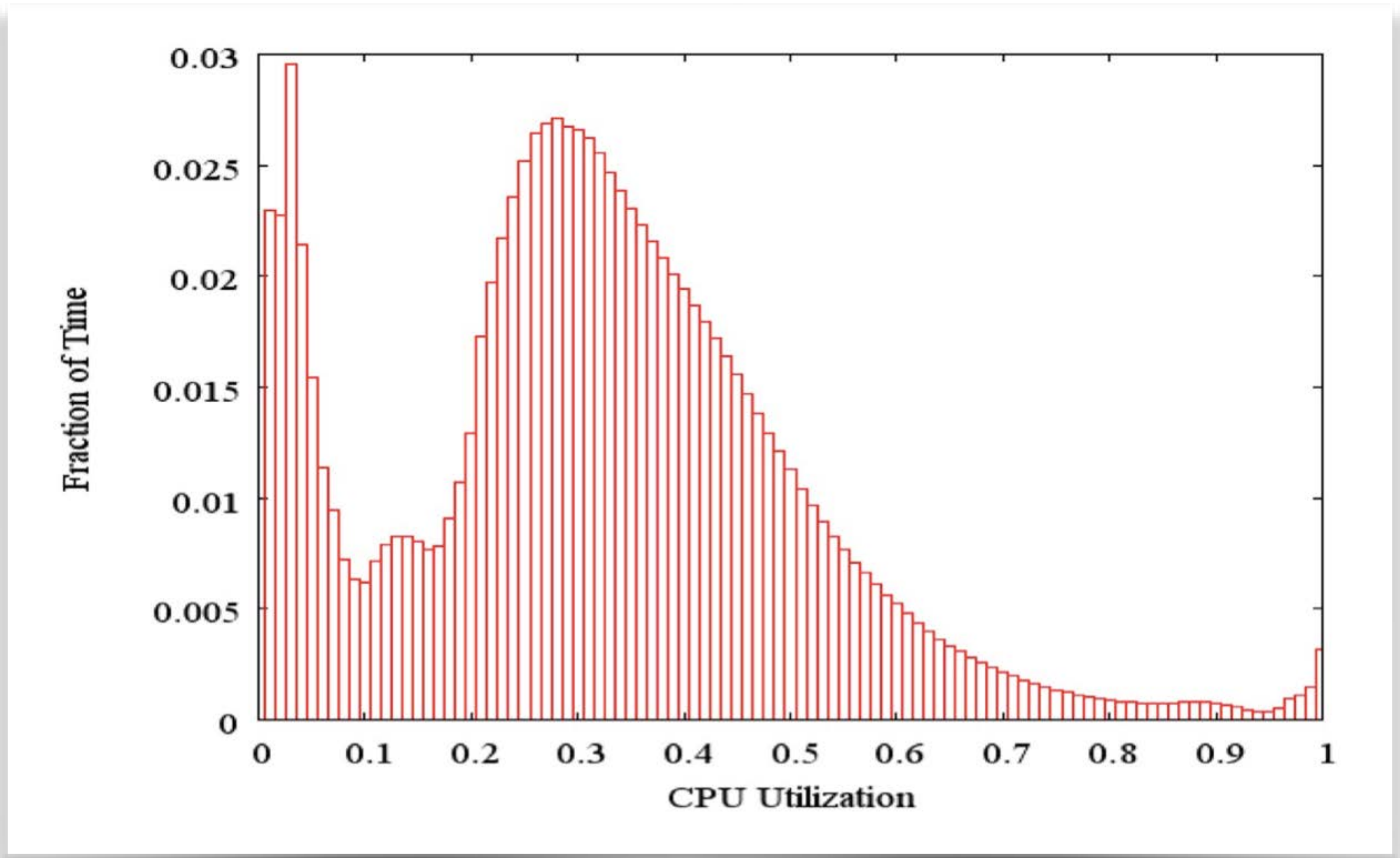
Sadly, most of this energy is wasted

Facebook in Luleå, Sweden will consume 1 TWh/year – approx 40.000 family houses

Problems with the Cloud

- Low level of determinism and predictability
  - No hard performance guarantees
- Data centers consume a lot of energy
- Data centers have low utilization

Datacenter Utilization



5000 google servers over 6 months

Source: “The Datacenter as a computer”, Barroso et al

Problems with the Cloud

- Low level of determinism and predictability
  - No hard performance guarantees
- Data centers consume a lot of energy
- Data centers have low utilization

➡ Room for better resource management techniques

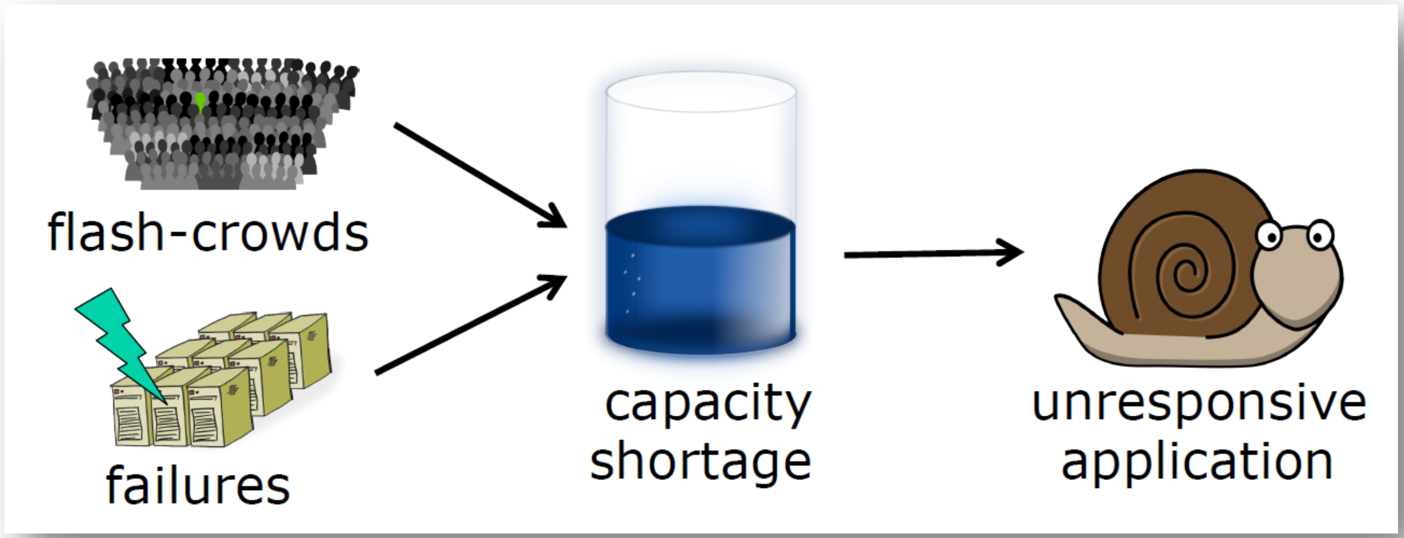
Content

- Motivation and Background
  - A simple queue length control example
- Resource Management for Multi-Core Embedded Systems
  - The ACTORS Resource Manager
    - Video demo
  - Game-Theory Resource Manager
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications

From Embedded to the Cloud

- Apply the game-theoretic resource manager to cloud applications
- Focus on state-free, request-based applications

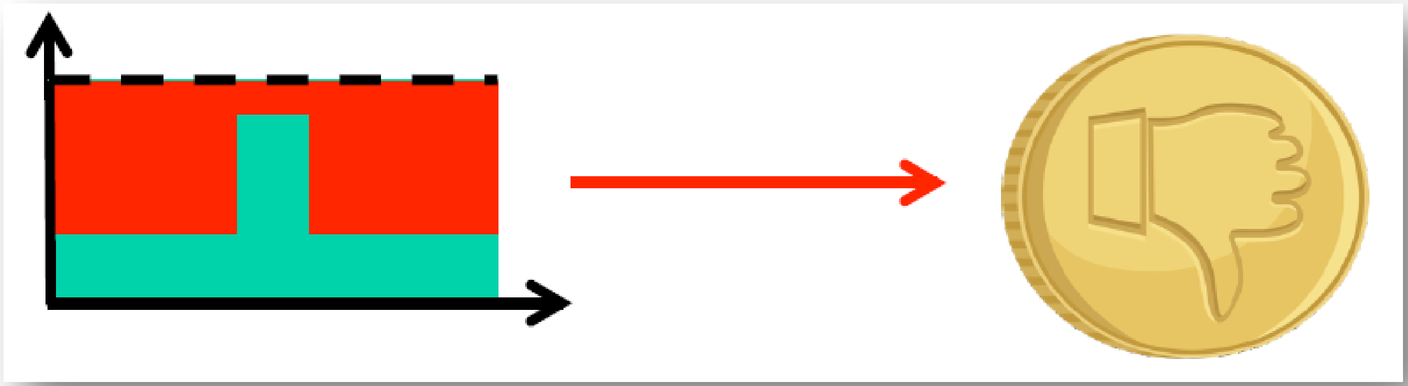
Problem: Unexpected Events



- 25% of end-users leave if load time > 4s*
- 1% reduced sale per 100ms load time*
- 20% reduced income if 0.5s longer load time**

* Amazon ** Google

Standard Practice



- Overprovisioning
  - Economically impractical

Brownouts

- We borrow from the concept of **brownouts** in power grids
- A brownout-compliant application can degrade user performance when needed to face unexpected conditions

Brownout

A brownout is an intentional or unintentional drop in voltage in an electrical power supply system. Intentional brownouts are used for load reduction in emergency conditions.

Brownout

A brownout is an intentional or unintentional drop in voltage in an electrical power supply system. Intentional brownouts are used for load reduction in emergency conditions.

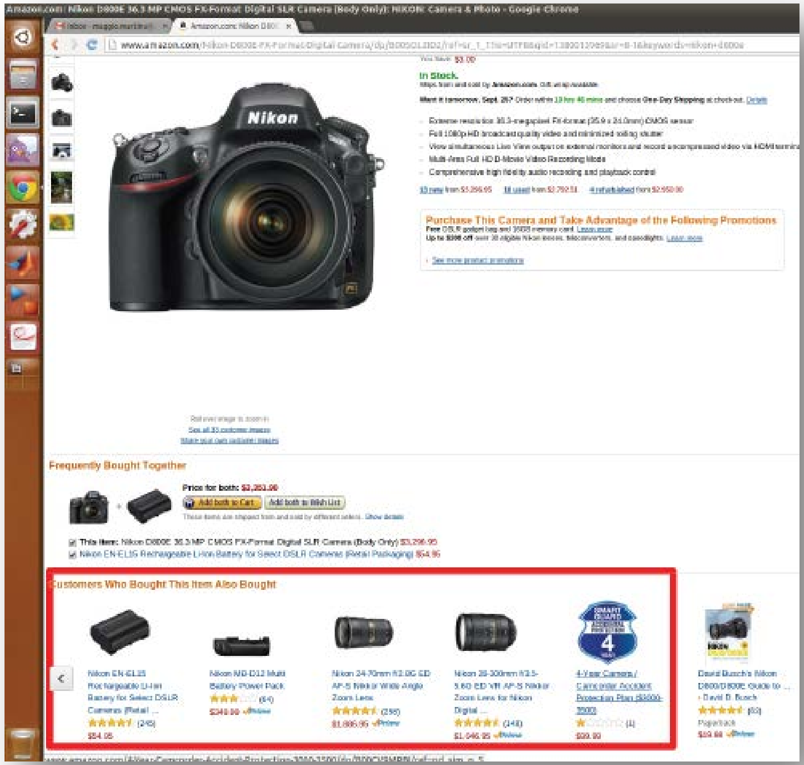
Tokyo Tower Brownout

Brownouts

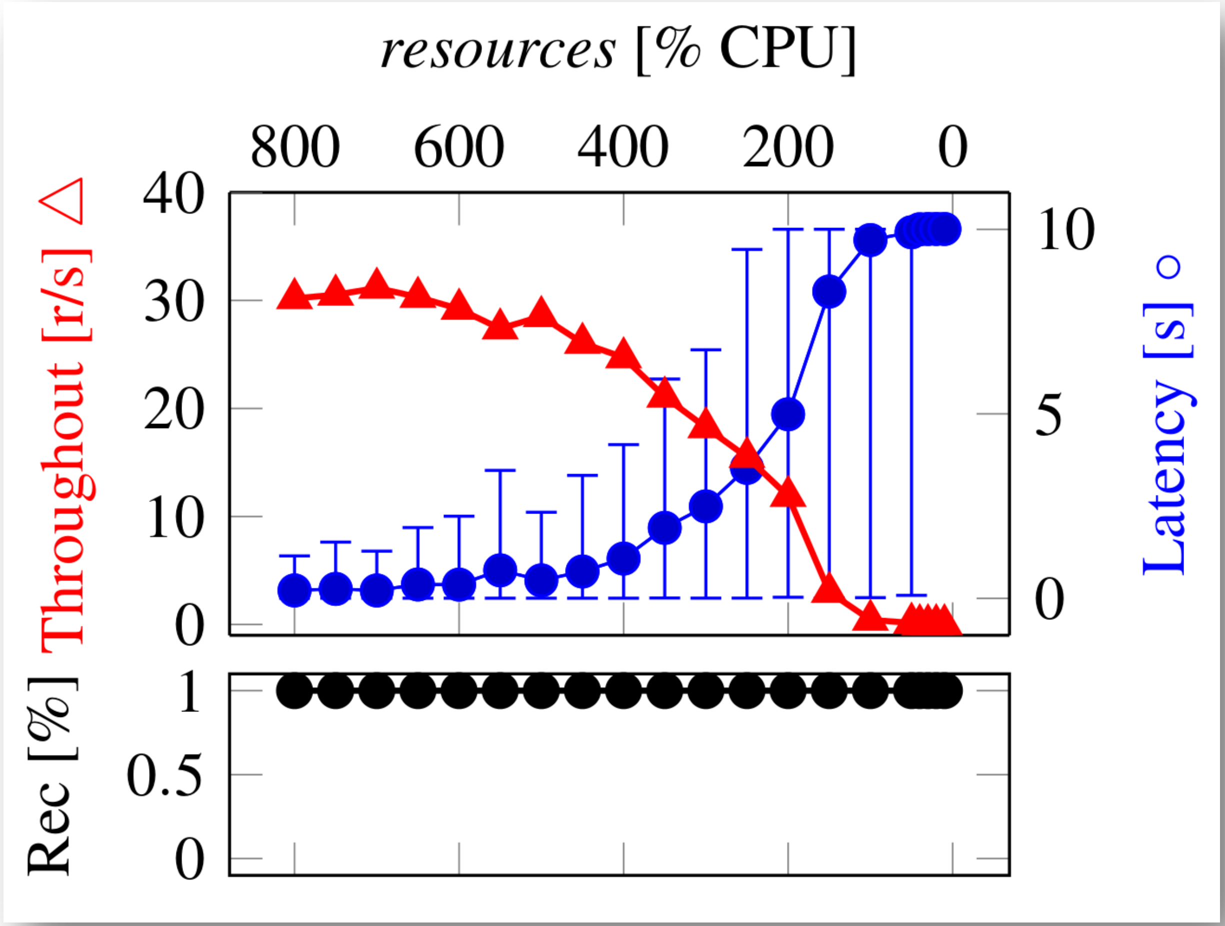
- We borrow from the concept of **brownouts** in power grids
- A brownout-compliant application can degrade user performance when needed to face unexpected conditions
- We assume that the reply to a request consists of a mandatory and an optional part
- During overload and/or lack of resources the percentage of requests that also receives the optional part can be decreased

Brownout Examples

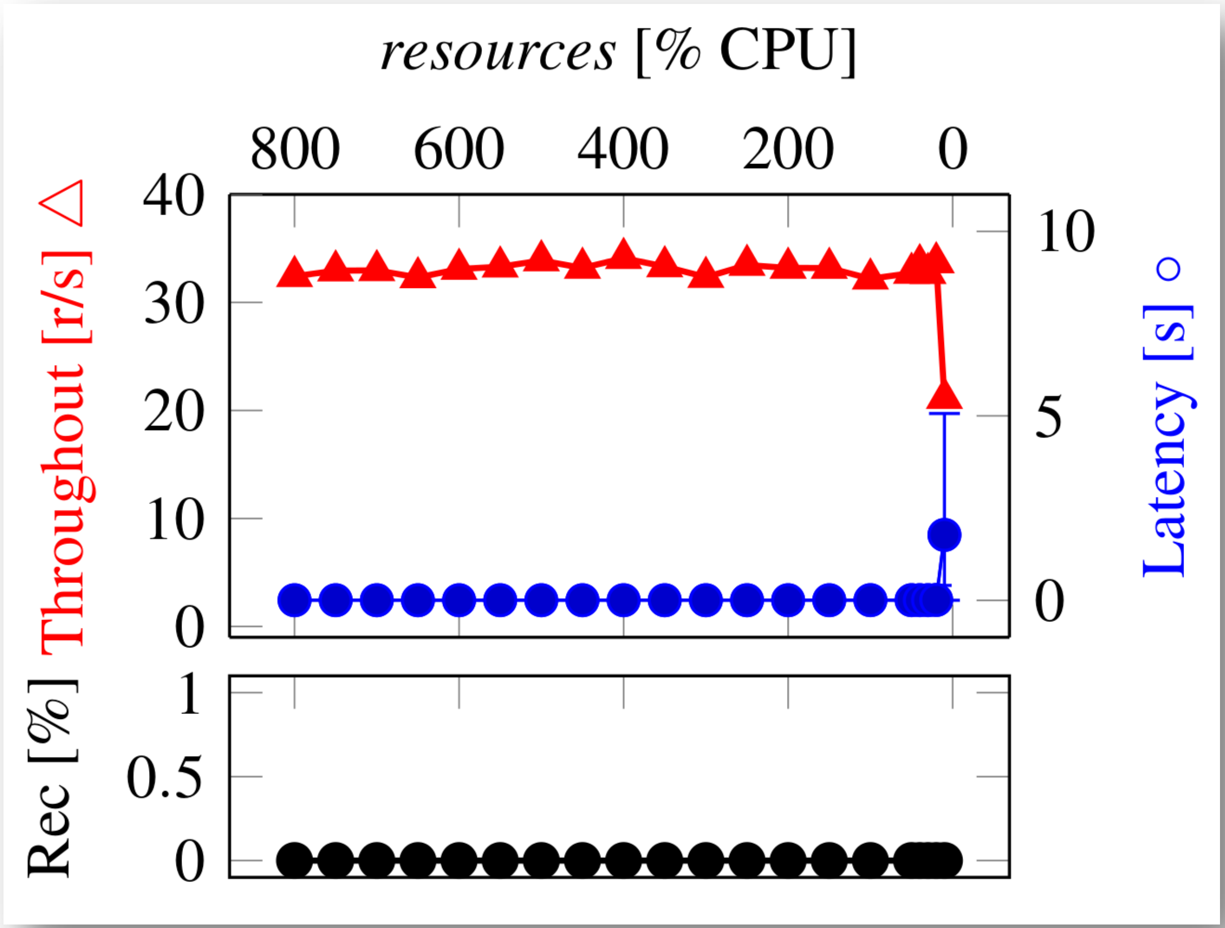
- E-commerce systems with recommendations
- Content adaptation in web server applications
- ……


In Stock.
Purchase This Camera and Take Advantage of the Following Promotions
Frequently Bought Together
Customers Who Bought This Item Also Bought

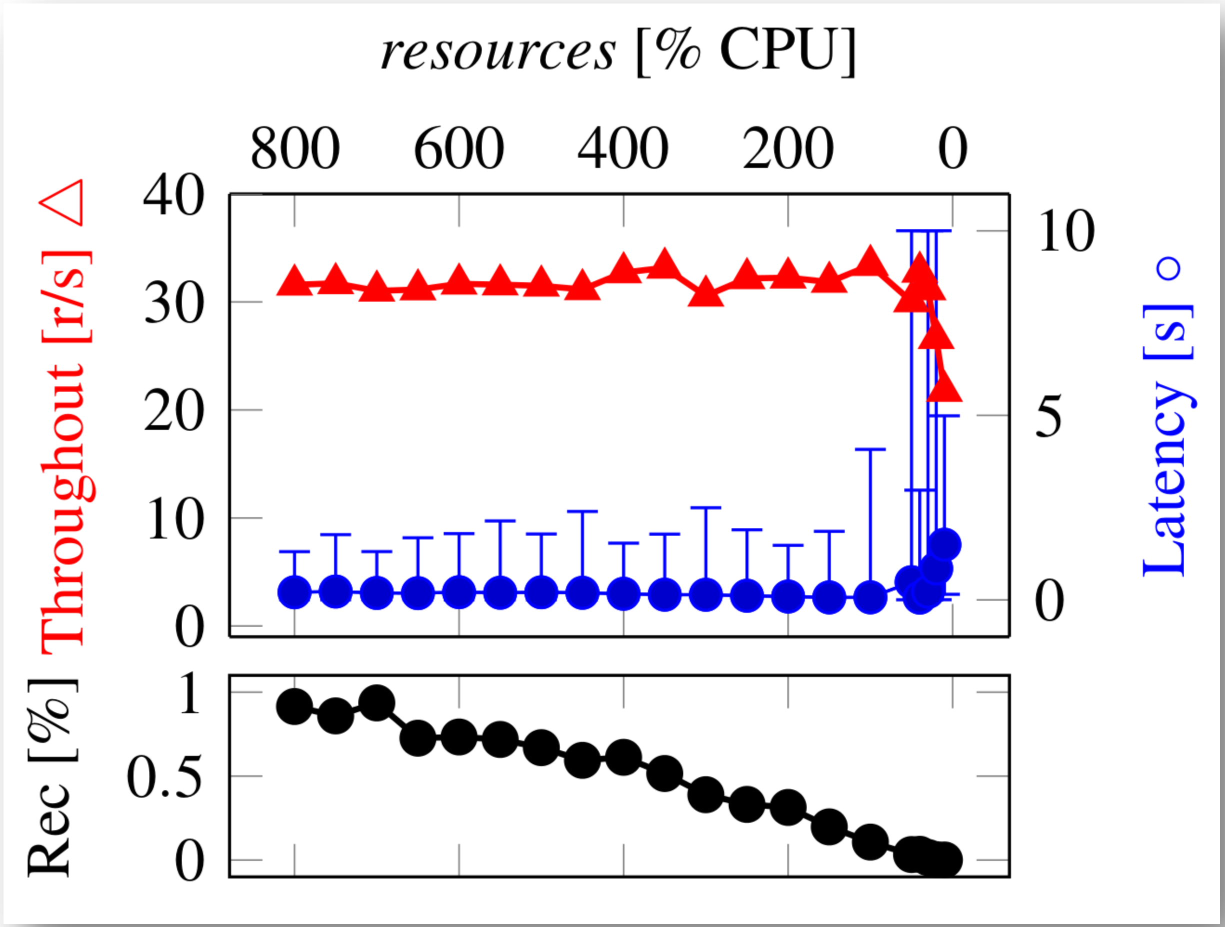
Always Service Optional Part



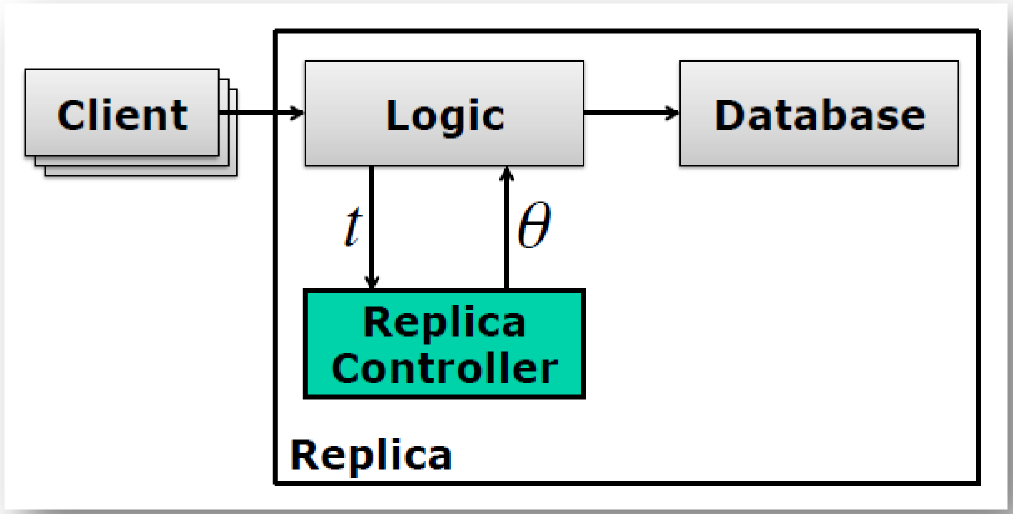
Never Service Optional Part



With Brownout



Brownout Control Loop

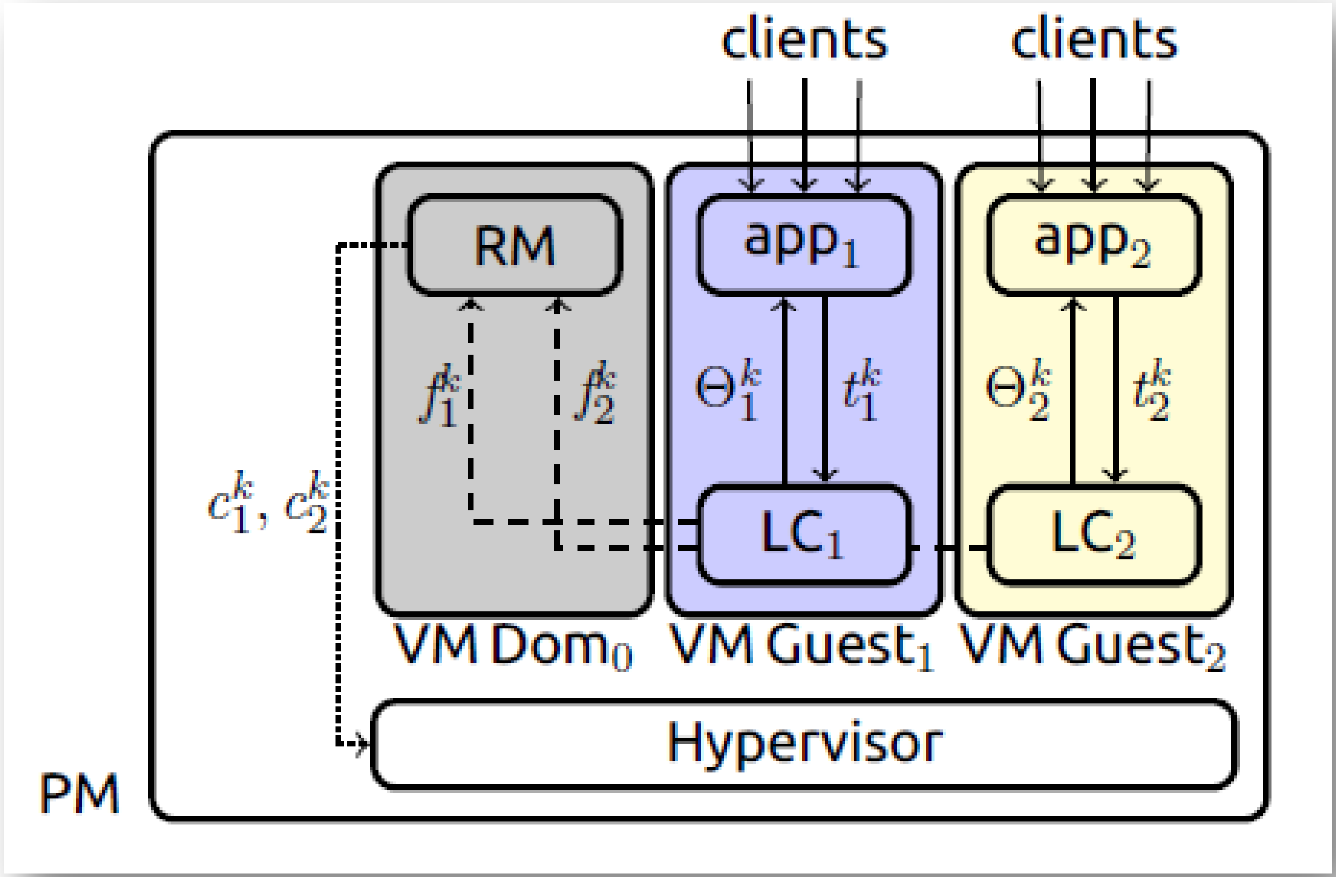


- Measured variable: Response time (average or 95% percentile)
- Control signal: Dimmer value (percentage of requests for which also the optional parts are calculated)
- Setpoint: 2 seconds

Brownout Controllers

- Adaptive PI and/or PID controllers
- Adaptive deadbeat controller
- Feedforward + feedback controller
  - Assumes knowledge of average service time for mandatory and optional parts

Brownout Resource Management



Resource Management Details

- Applications sends value of **matching function**

$$f_i^k = 1 - t_i^k / \bar{t}_i$$

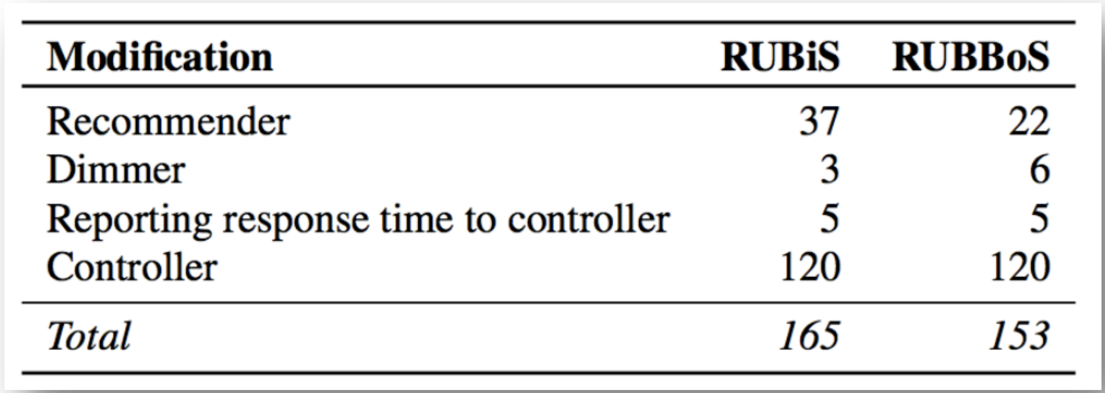
- Resource manager computes size of virtual machine

$$c_i^{k+1} = c_i^k - \varepsilon_{rm} \left(f_i^k - c_i^k \cdot \sum_p f_p^k \right)$$

- Proven to be fair and converge using game theory

Evaluation

- Applications
  - RUBiS: eBay-like prototype auction website
    - Added a recommender
  - RUBBoS: Slashdot-like bulletin board website
    - Added a recommender
    - Marked comments as optional
  - Effort in lines of code:



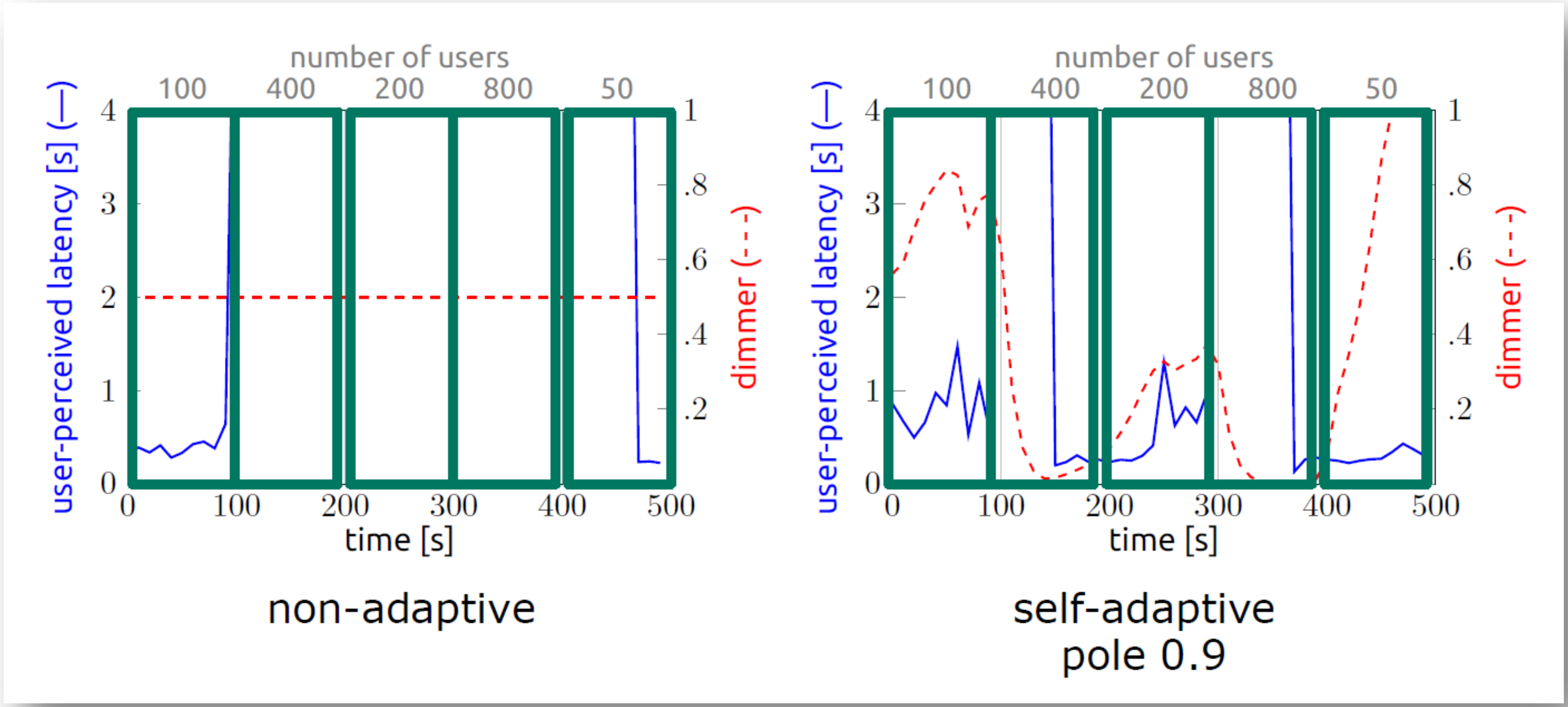
Modification	**RUBiS**	**RUBBoS**
Recommender	37	22
Dimmer	3	6
Reporting response time to controller	5	5
Controller	120	120
Total	*165*	*153*

Evaluation

- Hardware
  - One physical machine
  - Two AMD Opteron 6272 processors
  - 16 cores / processor
- Hypervisor
  - Xen
  - Each VM contains Apache web server, PHP interpreter, MySQL server, and brownout controller
- Client load generator
  - Custom built – httpmon
  - Closed loop model
    - Clients with think time

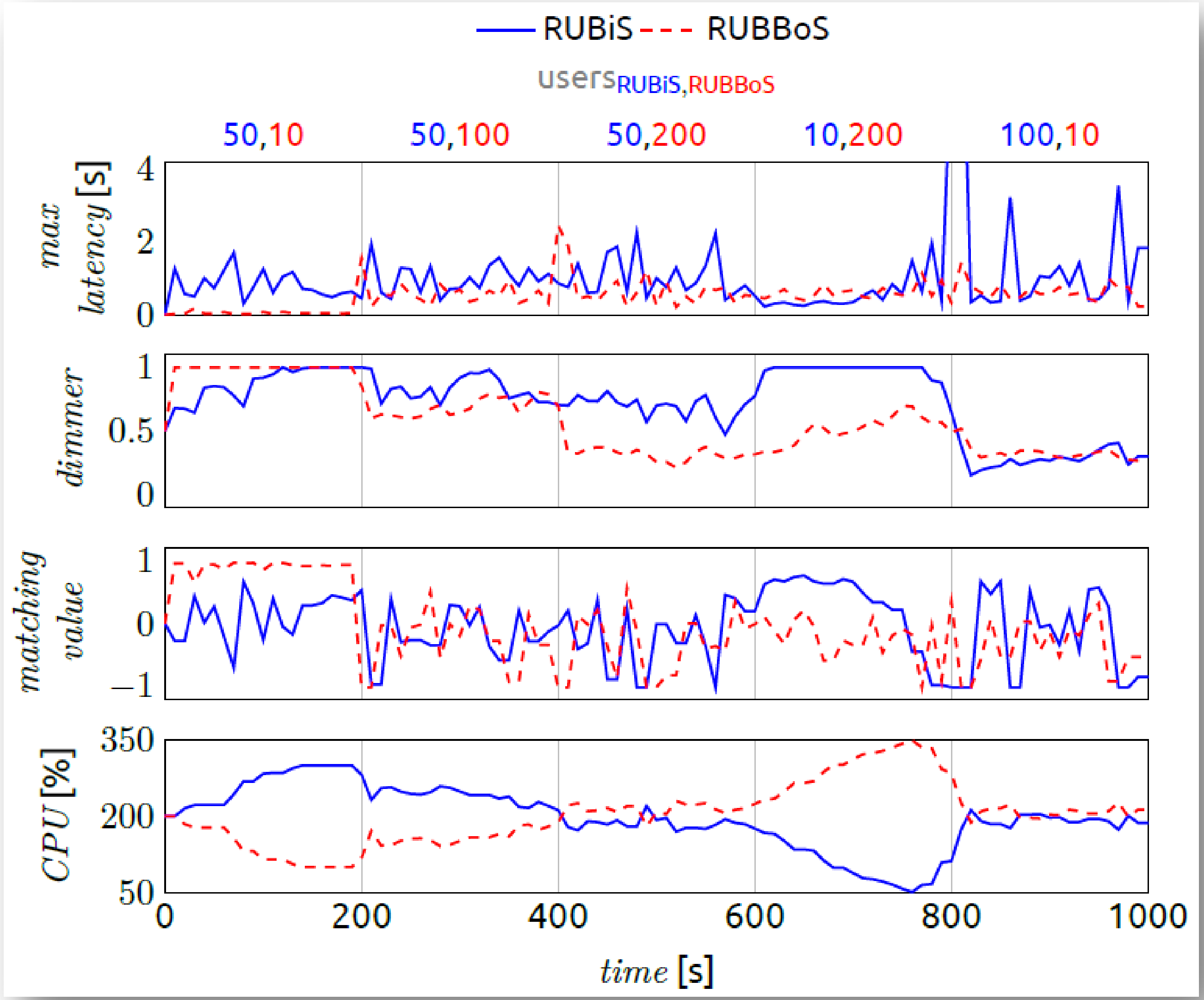
Experimental Results

- RUBiS flash crowds



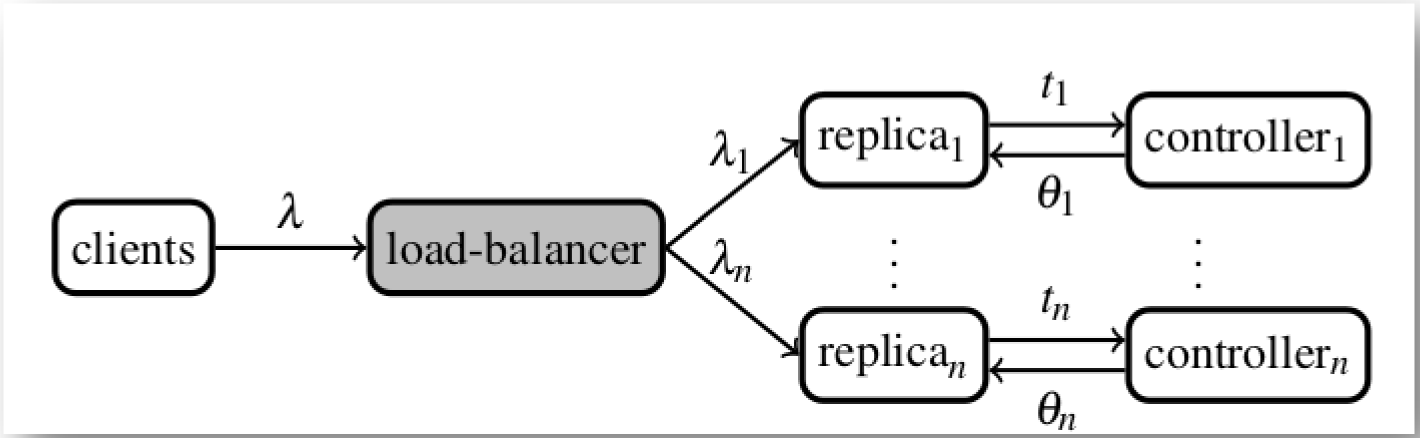
Experimental Results

Four cores used



Load Balancing

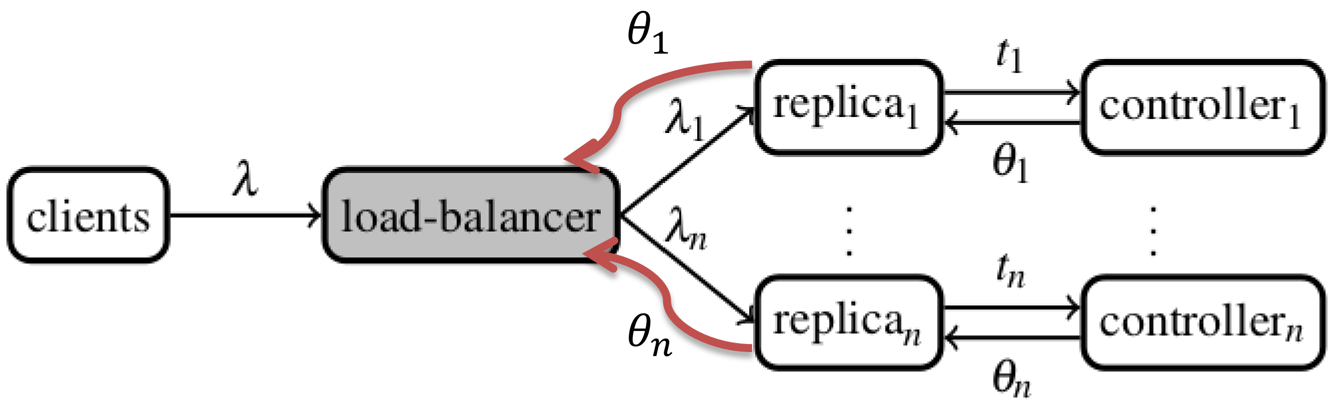
- Multiple replicas
  - Support scaling
  - Robust towards faults
- Load balancing



Existing Load Balancers

- D**ynamic** load balancers often measure response times and decide based on that
  - FRF, FRF-EWMA, 2RC, Predictive ...
- Does not work well in the presence of brownout control
  - the replica controllers keep the response times close to the setpoint

Brownout-Aware Load Balancing

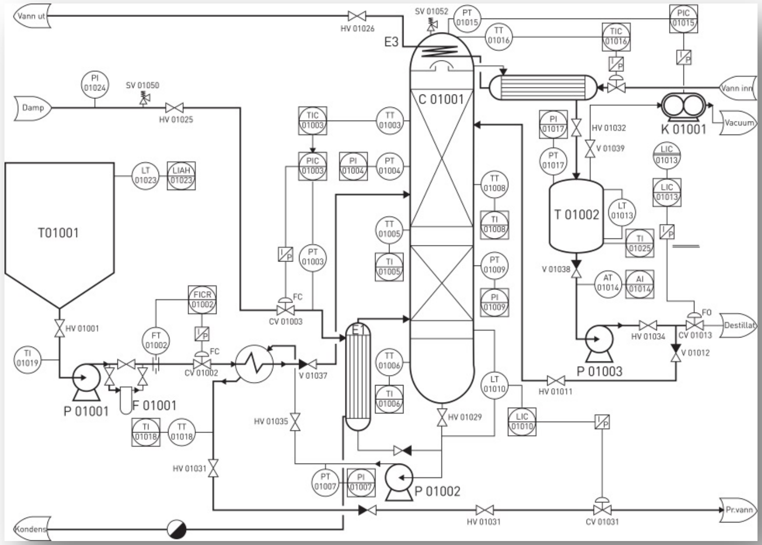


- New load balancing schemes have been developed that make use of the dimmer values
- Piggyback the dimmer values to the replies going back to the load balancer

Challenges

- Modeling of resources and of workload
  - Very difficult to predict peaks
- Scalability
  - What work for 500 servers will most likely not work for 50 000 servers
- Everything will break down all the time
  - Netflix chaos monkey
- Difficult to isolate the phenomenon under study
- Only industry have real data centers
  - Very industry-driven area
  - Google, Microsoft, …..

Process Automation Comparison



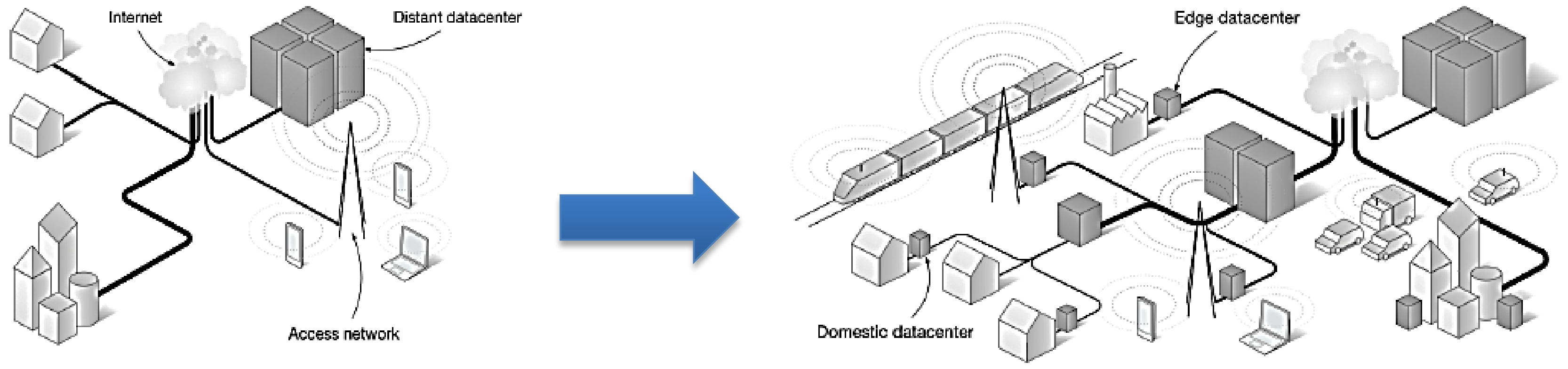
- Compare with process industry in the 1940-50s
  - Manual control
- Next step PI(D) +
  - Feedforward, cascade, ratio selector logic, split-range, mid-range, ……
  - Controller patterns
    - Flow control, temperature control, …..
- Now
  - MPC control + optimization-based long-term planning
  - But still PI(D) at the lowest level

Process Automation Comparison

- Lessons:
  - Keep it simple (”KISS”)
  - Start with classical, e.g. PID, control techniques
  - Add optimization control on top of this if required
- Differences:
  - Flexibility and generality
  - Behavior not governed by laws of nature
- Maybe one have to constrain the flexibility in order to achieve determinism

The Future Distributed Cloud

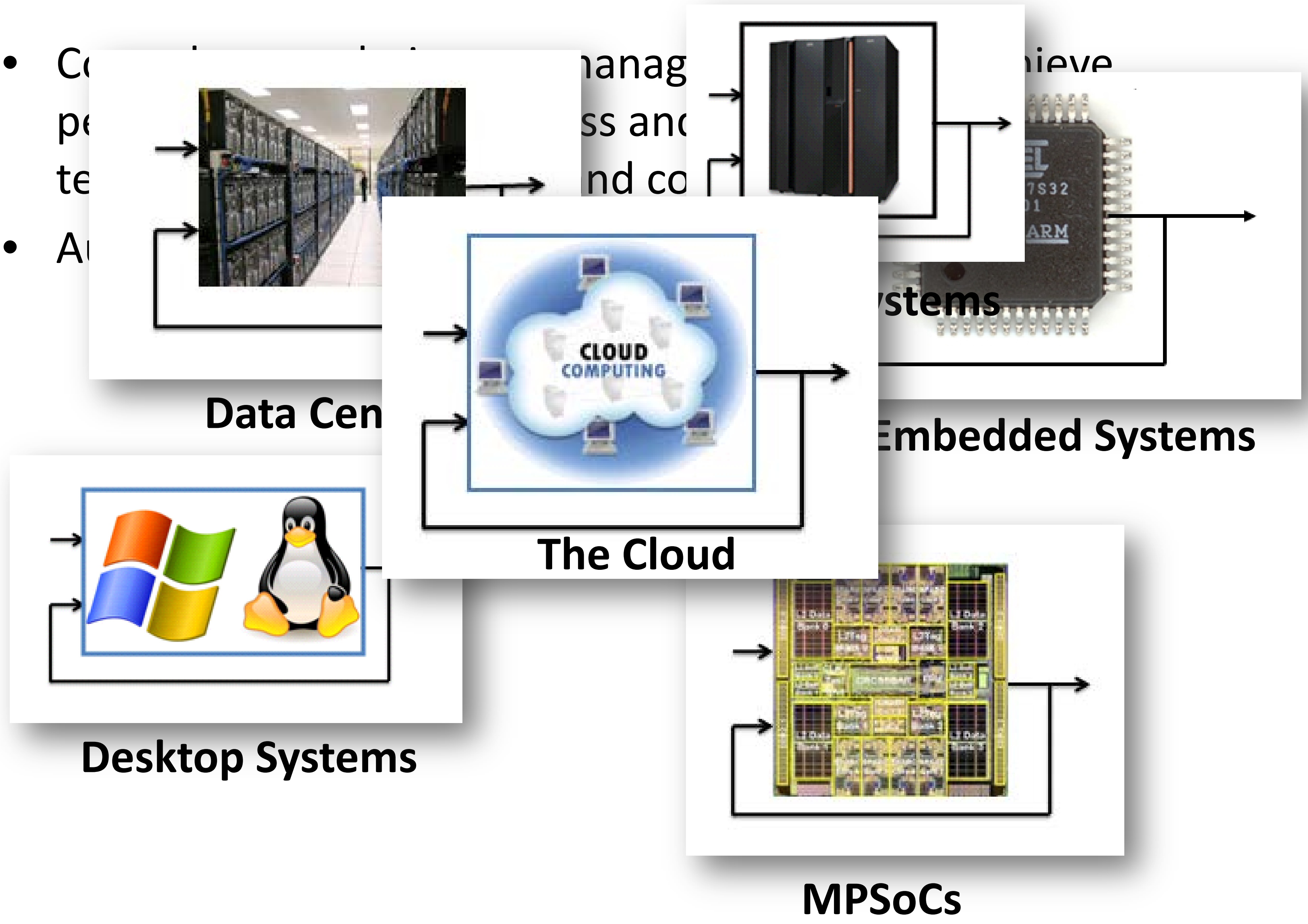
- 5G opens up for new application classes
  - Mission-critical applications, e.g., closed loop control
  - Requirements on low latency and high availability –> predictability
- Network and cloud convergence
  - Boundary between the network and datacenters disappears
  - Telco cloud, mobile cloud, infinite cloud
  - Computations to be dynamically deployed in all types of nodes, incl base stations and remote data centers
  - The Resource Management Problem: To decide how much and what type of resources to allocate and where and when to deploy them.



Today Future

Feedback Computing

- Co nanag ieve pe ss anc te nd co
- A



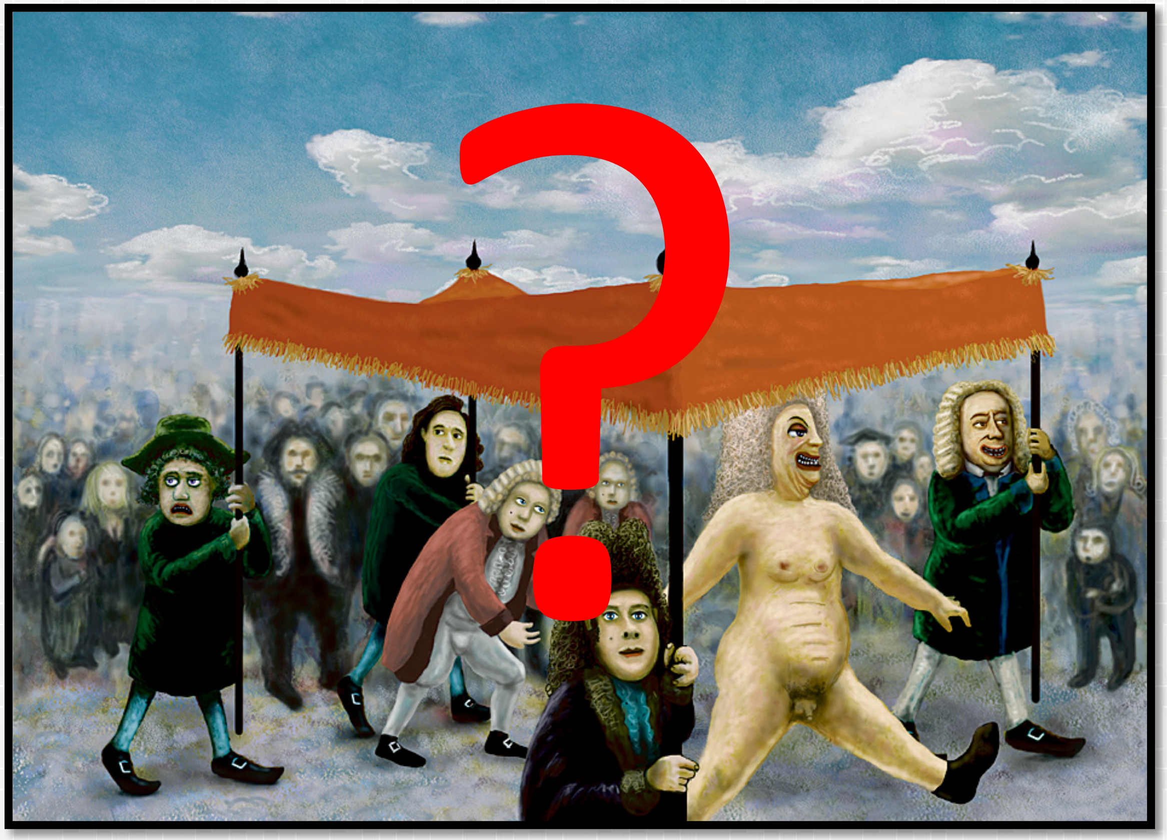
Content

- Cyber-Physical Systems – My Personal View
- Control of Computer Systems
  - Motivation and Background
  - A simple queue length control example
- Resource Management for Multi-Core Embedded Systems
  - The ACTORS Resource Manager
    - Video demo
  - Game-Theory Resource Manager
- The Cloud
  - Problems and Challenges
  - Brownout-inspired resource management for web-service applications
- **A CPS Story**

CPS Software: Emperor’s New Clothes or Not?

An Embedded Controls Point of View

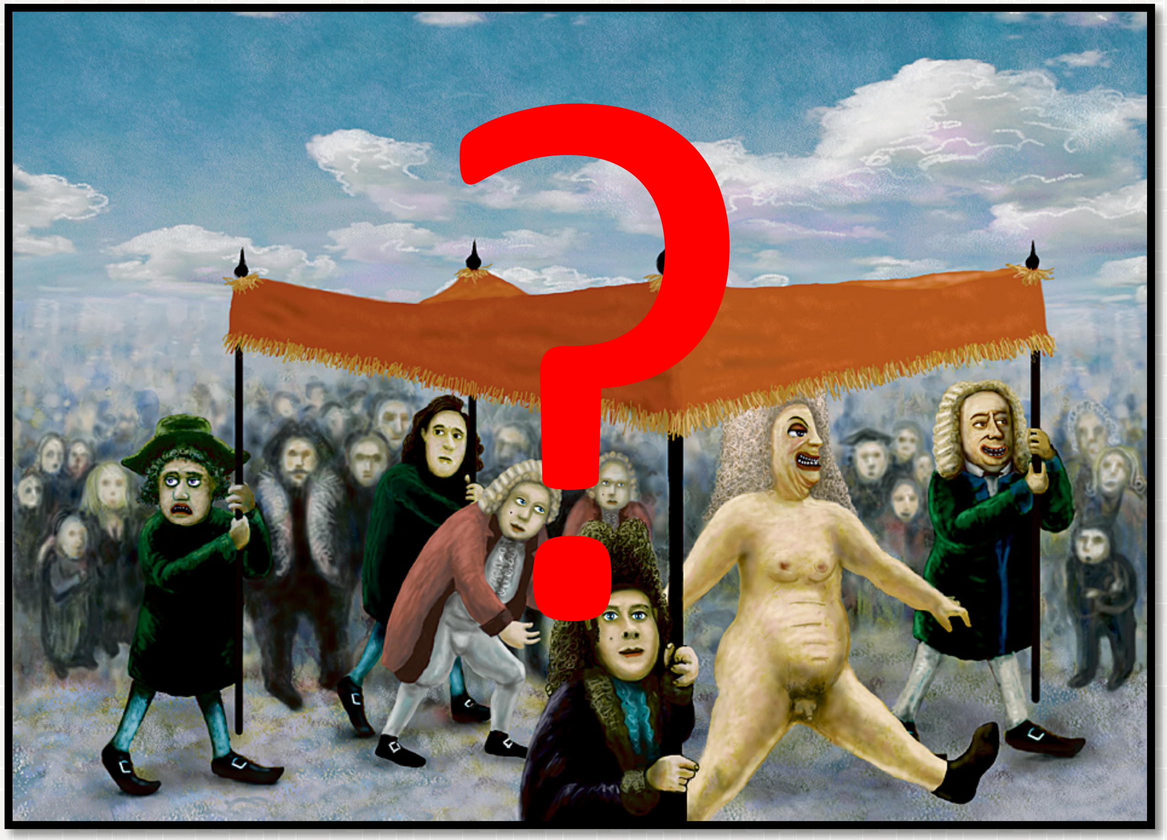
Karl-Erik Årzén, Lund University, Sweden



HC Andersen

CPS Software: Emperor's New Clothes or Not?

An Embedded Controls Point of View



Emperor’s New Clothes - YES

- Close interaction with the physical environment has always been crucial both in embedded systems and control
- Same thing with limited computing resources
- Networks is also nothing new
  - Distributed embedded systems
  - NoC
  - Networked Control
  - ……..
- CPS is mostly a matter of scale

Emperor's New Clothes - NO

- Traditional embedded systems
  - Too simplified view of physical part
  - Generator of events with deadlines
    - Periodic, aperiodic, ……
  - Too strong emphasis on worst-case design and hard guarantees
- Traditional control
  - Too simplified view of cyber part
    - Periodic execution
    - Zero or constant latencies
    - Parallel

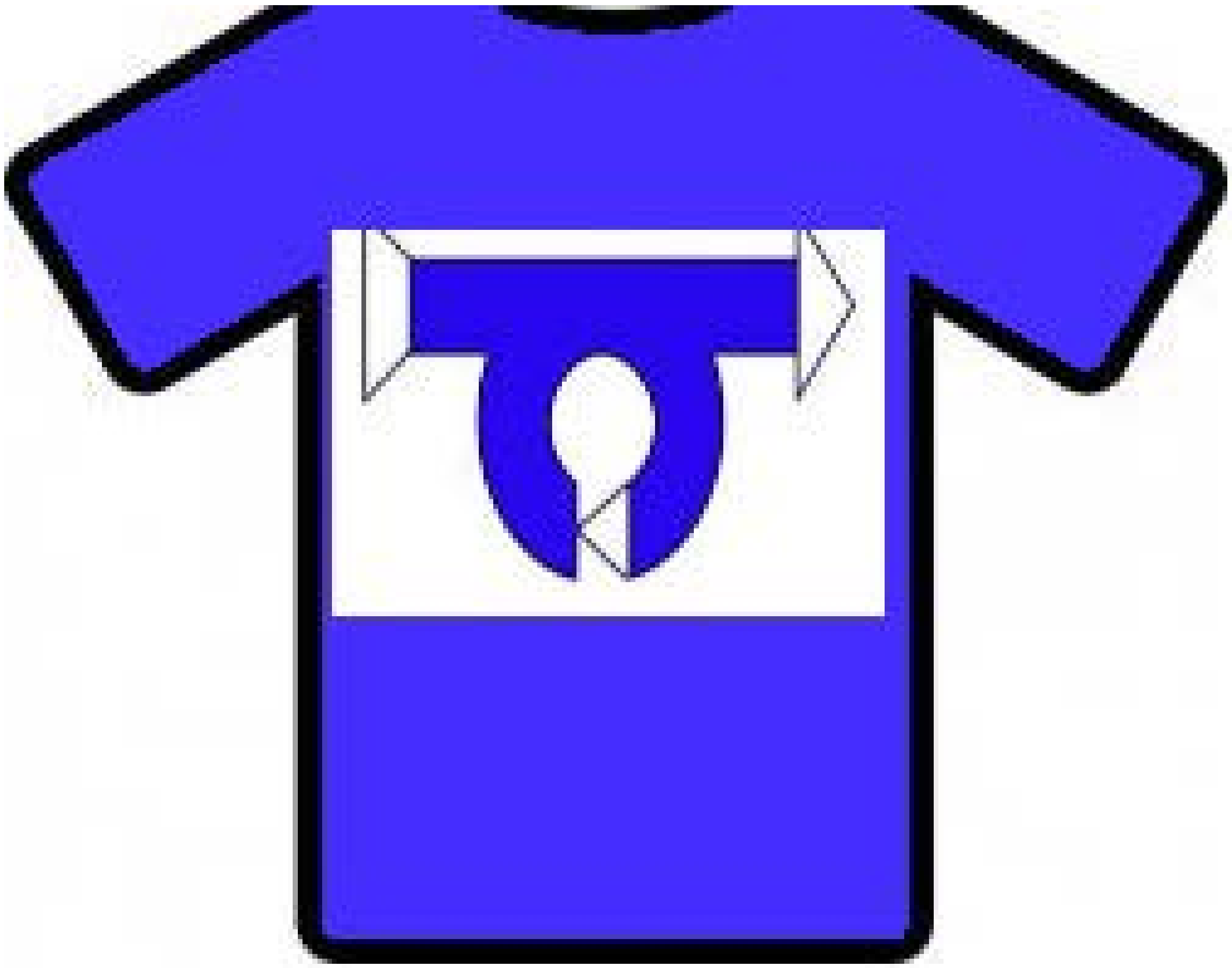
The Emperor's New Clothes – CPS Version

A King

Or Queen

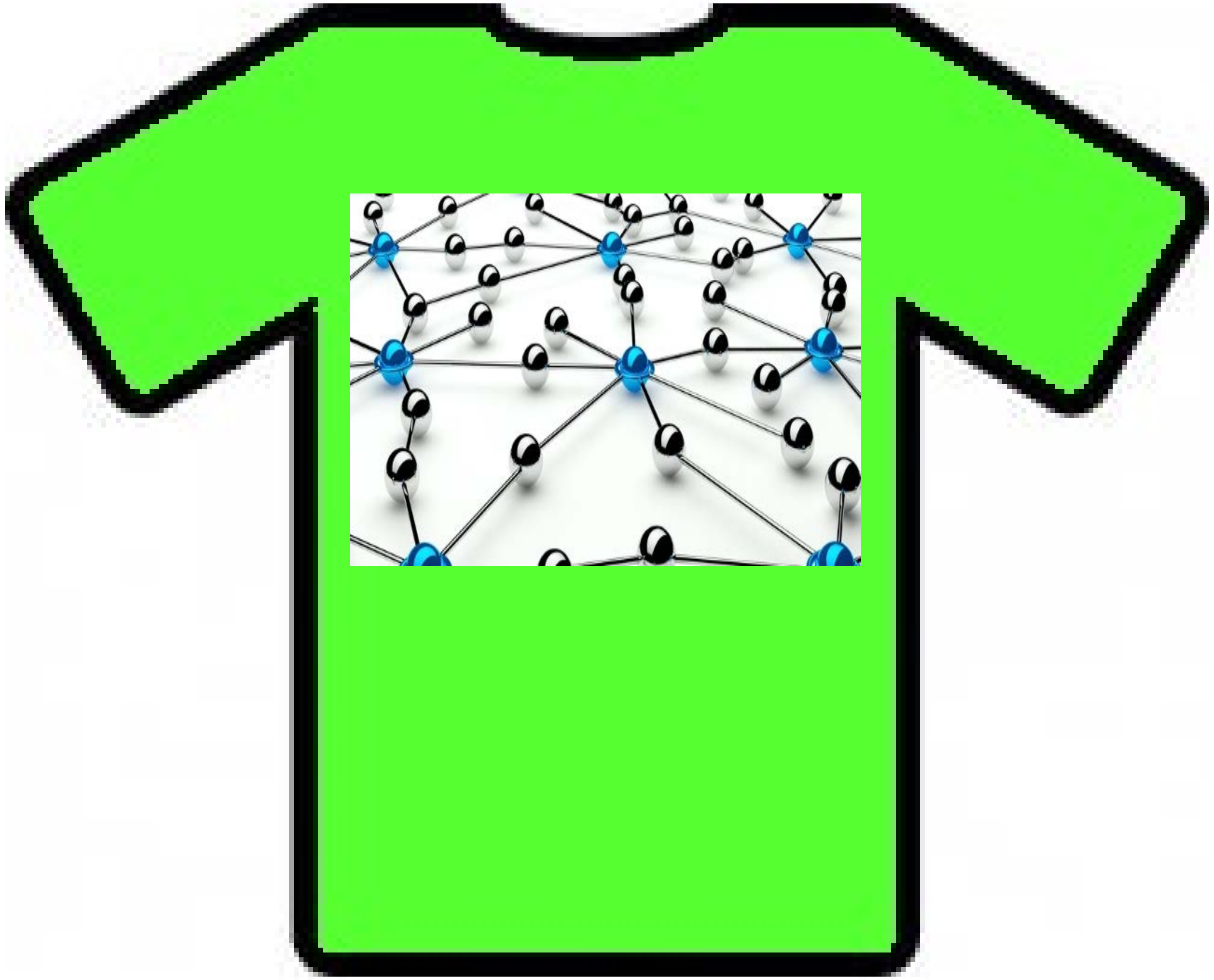
Embedded Systems Coat

Control Coat



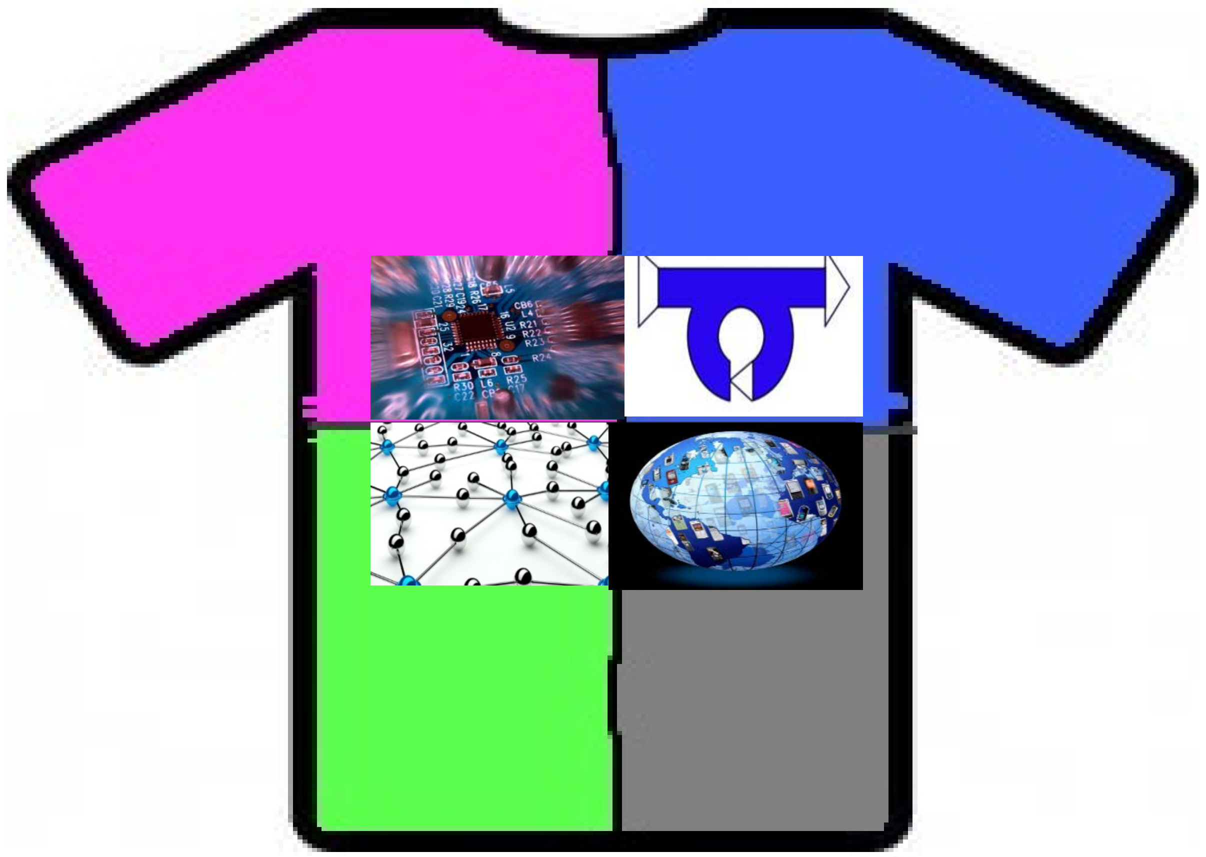
Internet of Things Coat

System-of-Systems Coat



CPS Coat

The Universal Coat of All Colors



Questions?

Further Information

- ACTORS Resource Manager
  - E. Bini, G. Buttazzo, J. Eker, S. Schorr, R. Guerra, G. Fohler, K-E. Årzén, V. Romero, C. Scordino: "Resource Management on Multi-core Systems: the ACTORS approach“, **IEEE Micro**, 31:3, pp. 72-81, May 2011
- Game-Theoretical Resource Manager
  - Georgios Chasparis, Martina Maggio, Karl-Erik Årzén, Enrico Bini: "Distributed Management of CPU Resources for Time-Sensitive Applications". In 2013 **American Control Conference**, Washington DC, USA, June 2013
  - Martina Maggio, Enrico Bini, Georgios Chasparis, Karl-Erik Årzén: "A Game-Theoretic Resource Manager for RT Applications". In 25th Euromicro Conference on Real-Time Systems, **ECRTS13**, Paris, France, July 2013
  - Georgios Chasparis, Martina Maggio, Enrico Bini, Karl-Erik Årzén: “Design and Implementation of Distributed Resource Management for Time Sensitive Applications”, Accepted for publication in **Automatica**

Further Information

Cloud Brownout

- Cristian Klein, Martina Maggio, Karl-Erik Årzén, Francisco Hernández-Rodriguez: "Introducing Service-level Awareness in the Cloud". In 2013 **ACM Symposium on Cloud Computing**, Santa Clara, CA, October 2013
- Cristian Klein, Martina Maggio, Karl-Erik Årzén, Francisco Hernández-Rodriguez: "Brownout: Building more Robust Cloud Applications", 36th International Conference on Software Engineering (**ICSE**), Hyderabad, India, 2014
- Martina Maggio, Cristian Klein, Karl-Erik Årzén "Control strategies for predictable brownouts in cloud computing", **IFAC World Congress**, Cape Town, South Africa, August 2014
- Jonas Dürango, Manfred Dellkrantz, Martina Maggio, Cristian Klein, Alessandro Vittorio Papadopoulos, Francisco Hernández-Rodriguez, Erik Elmroth, Karl-Erik Årzén, "Control-theoretical load-balancing for cloud applications with brownout", **CDC** 2014
- Cristian Klein, Alessandro Vittorio Papadopoulos, Manfred Dellkrantz, Jonas Dürango, Martina Maggio, Karl-Erik Årzén, Francisco Hernández-Rodriguez, Erik Elmroth, "Improving Cloud Service Resilience using Brownout-Aware Load-Balancing", In 33rd IEEE Symposium on Reliable Distributed Systems (**SRDS**), 2014

Contributors

ACTORS:

- Johan Eker (Ericsson), Giorgio Buttazzo (SSSA), Enrico Bini (SSSA), Claudio Scordino (Evidence), Gerhard Fohler (TUKL), Stefan Schorr (TUKL), Vanessa Romero Segovia (LU), ……

Game-Theoretical RM:

- Martina Maggio (LU), Enrico Bini (LU/SSSA), Georgios Chasparis (LU/Software Competence Center Hagenberg)

Cloud Brownout:

- Martina Maggio, Alessandro Papadopoulos, Manfred Dellkrantz, Jonas Dürango (LU), Cristian Klein Umeå Univ), ...

So What Next?

- CPS, Systems of systems, IoT, Smart X, …. Has been around for some time
- What will be the next area?

Autonomous Systems

WASP | WALLENBERG AUTONOMOUS SYSTEMS PROGRAM

- 1.8 Billion SEK (200 million USD) over 10 years
- Four Swedish universities (Chalmers, KTH, Lund, Linköping)
- Knut and Alice Wallenberg Foundation
- Content:
  – Autonomous physical artefacts
    - Vehicles, robots, ……
  – Autonomous ”Cyber” Systems
  – Software for autonomous systems

WASP | WALLENBERG AUTONOMOUS SYSTEMS PROGRAM

- Open positions
  - 26 PhD student positions
  - 21 industrial PhD student positions
  - 9 senior recruitments
- www.wasp-sweden.se