Cloud Resource management and scheduling

ACKNOWLEDGEMENTS

- This presentation has been made from various sources with minimum modifications from the presenter.
- The presenter is grateful to the authors of those various sources.
- The presenter acknowledge the efforts of those authors and thank them wholeheartedly.

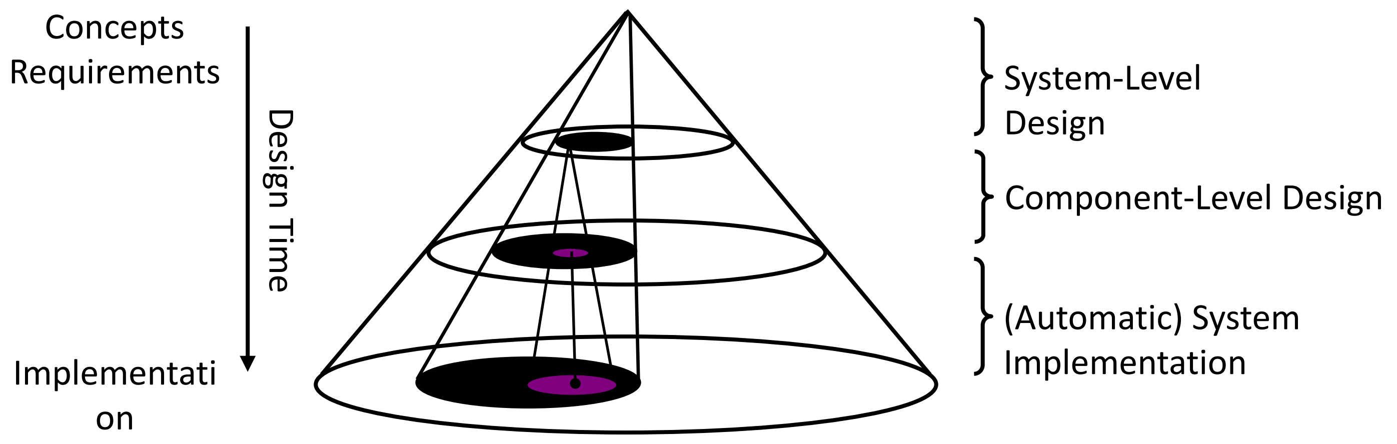
Some Basics

- Performance modeling
- **Control Theory**
- **Utility Theory**
- **Auction**
- Thresholding

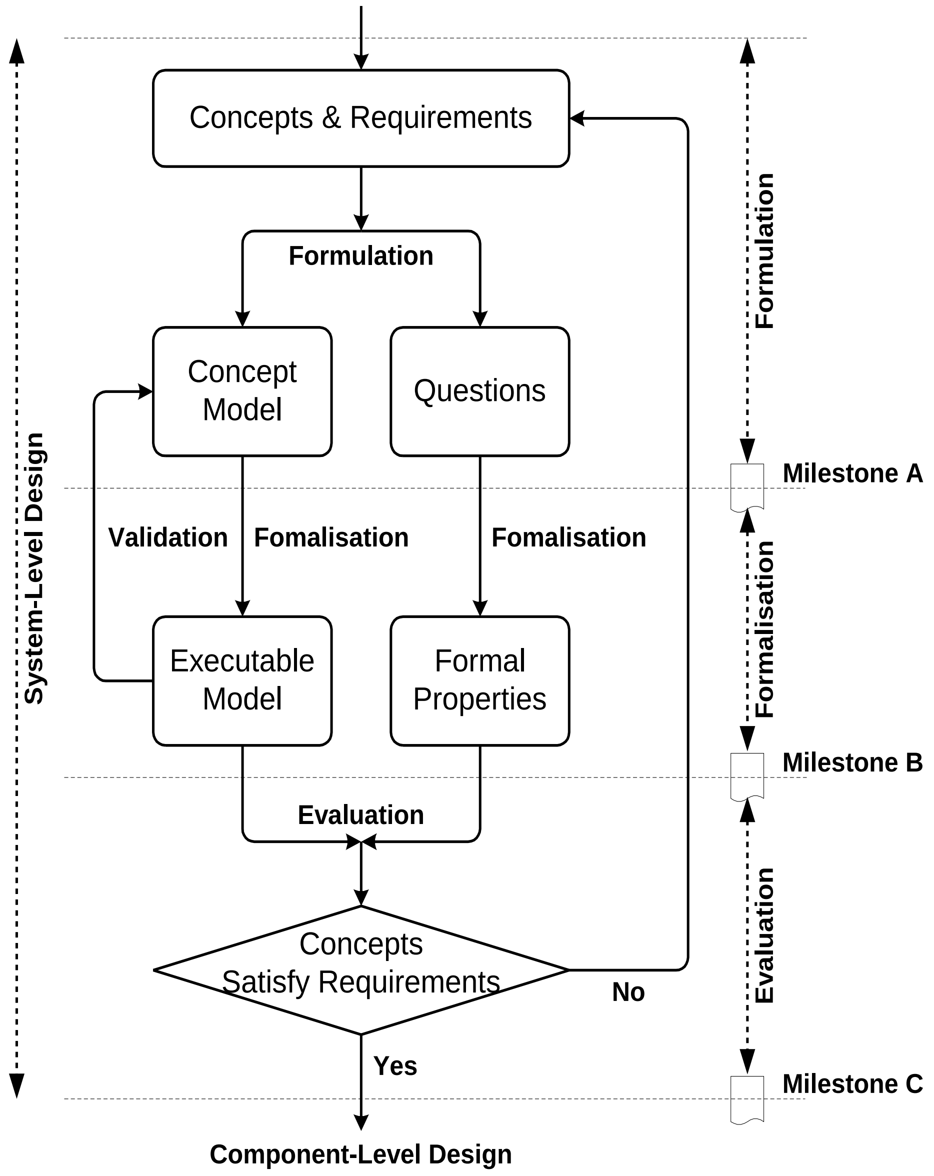
PERFORMANCE MODELLING

Introduction

- Designing a complex system within limited time involves taking important decisions in an early phase of the design process, which may have a deep impact on the performance of the system



- Early assessment of the impact of design decisions involves using system-level design methods and tools



Three Phases

- *Formulation*
  - Informal identification and specification of system concepts and design issues
- *Formalisation*
  - Formal specification of behaviour and architecture
  - Validation of adequacy of formal system specification
  - Formal specification of the properties to evaluate
- *Evaluation*
  - Property analysis and design decisions

Performance Modelling

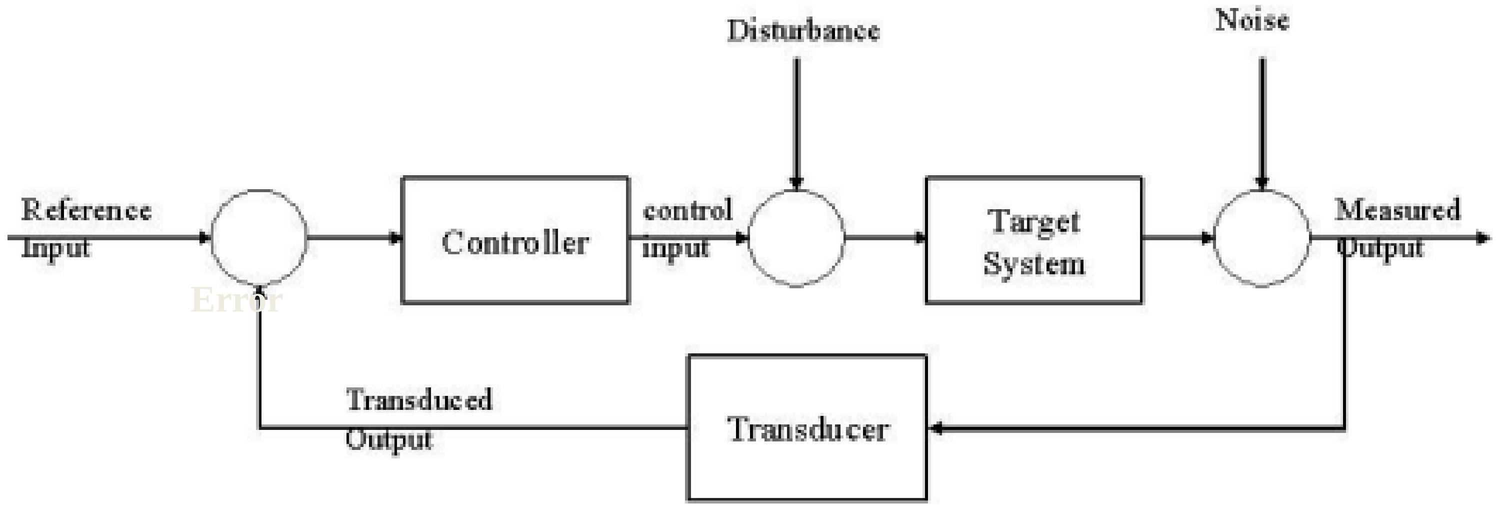
- Structured and repeatable approach to modeling the performance of your software.
- Generally ignored until there is a problem.
  - problems are frequently introduced early in the design
  - cannot always be fixed through tuning.
  - Fixing architectural or design issues later in the cycle is not always possible.

Performance modeling -benefits

- Performance becomes part of your design.
  - not an afterthought
- Modeling helps answer the question:
  - Will your design support your performance objectives?
  - evaluate your tradeoffs earlier
- You know explicitly *what design decisions* are *influenced by performance*

CONTROL THEORY

A Feedback Control System



<u>Components</u>

•Target system: what is controlled
•Controller: exercises control
•Transducer: translates measured outputs

<u>Data</u>

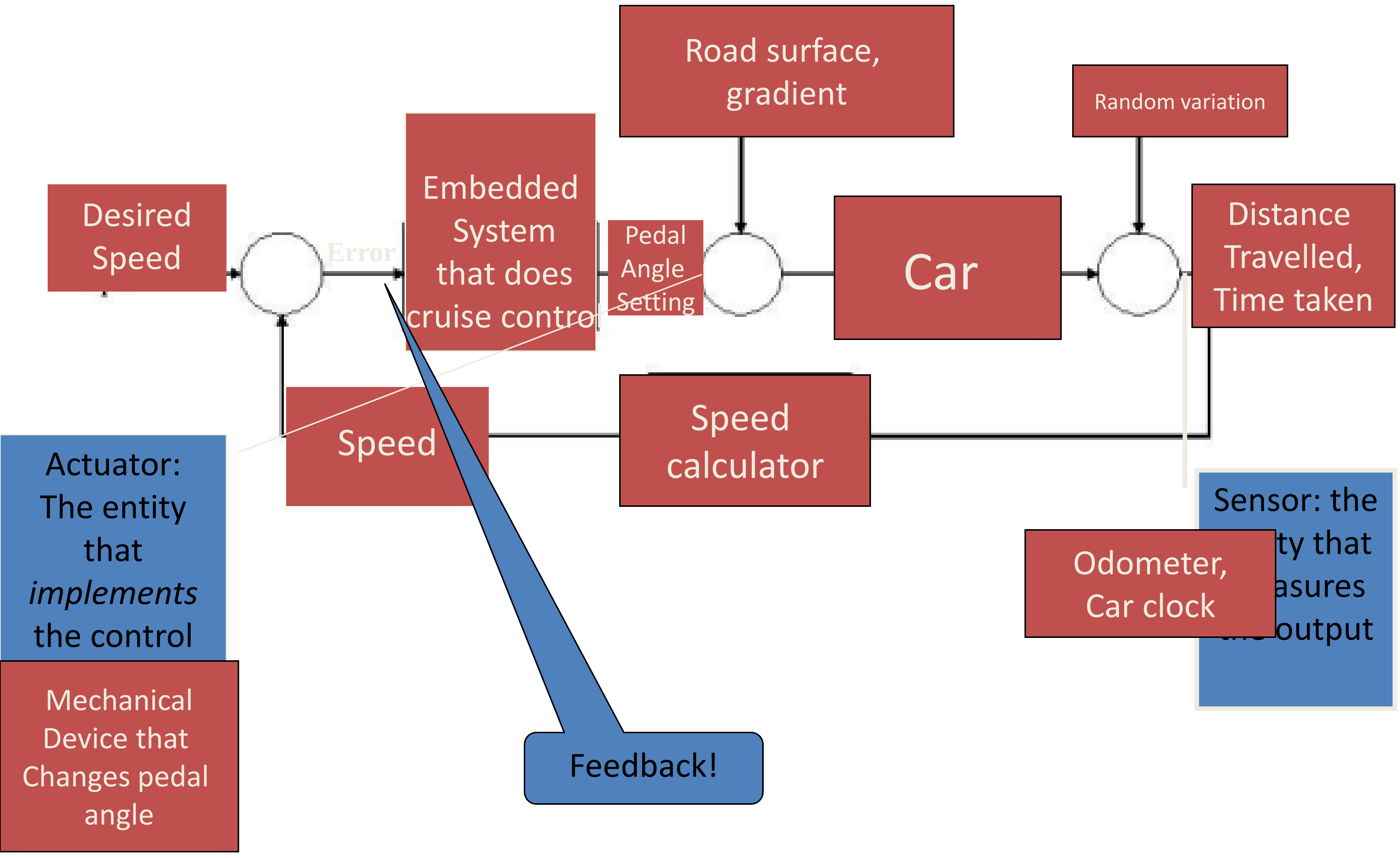
•Reference input: objective
•Control input: manipulated to affect output
•Disturbance input: other factors that affect the target system
•Transduced output: result of manipulation

Given target system, transducer, Control theory finds controller that adjusts control input to achieve given measured output in the presence of disturbances.

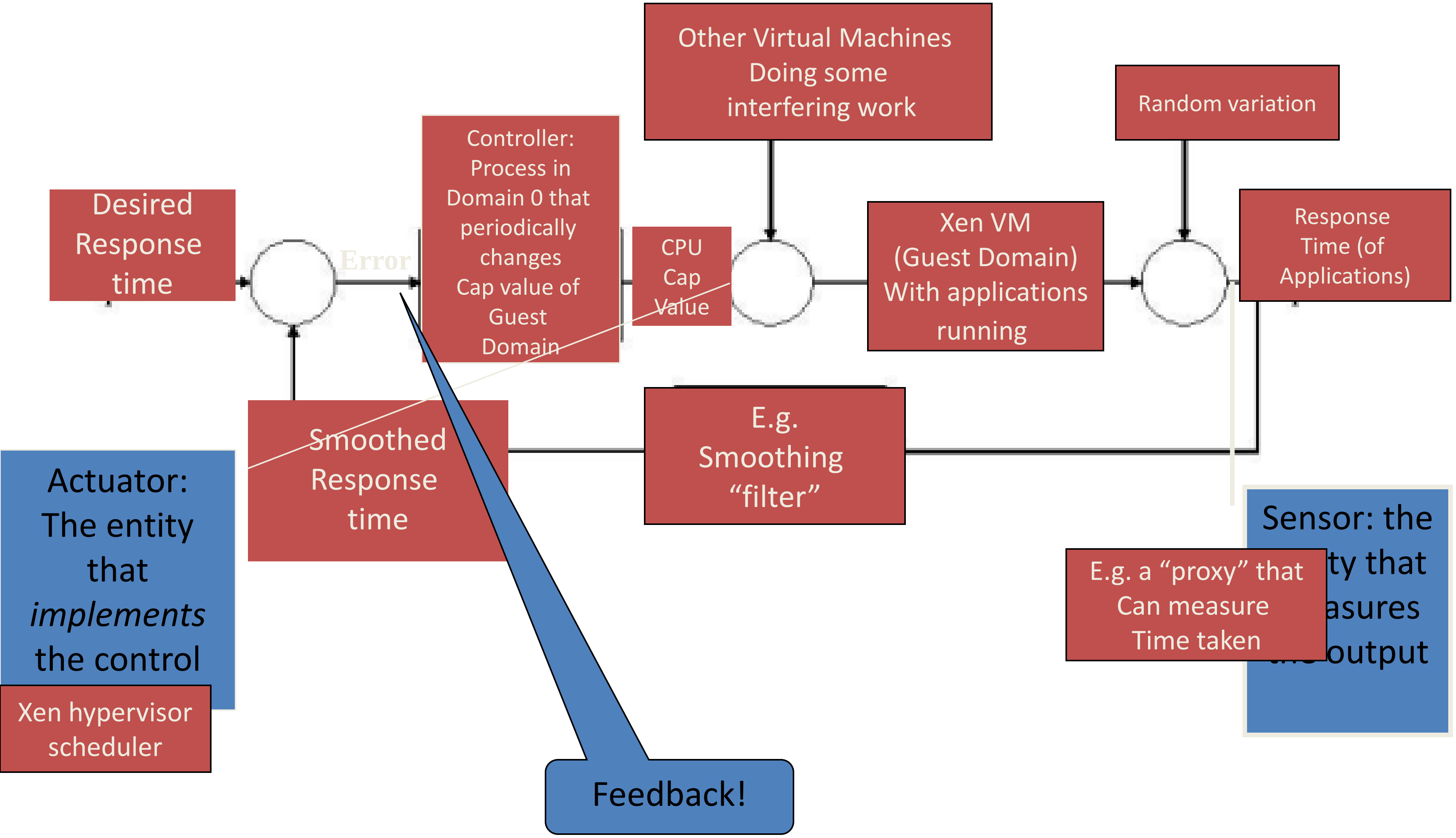
Control System Examples

- AC temperature control
  - Controlled Variable: temperature
  - Control variables: fan speed, time for which compressor is on?
- Car cruise control
  - Controlled variable: Car speed
  - Control variable: Angle of pushing of accelerator and brake

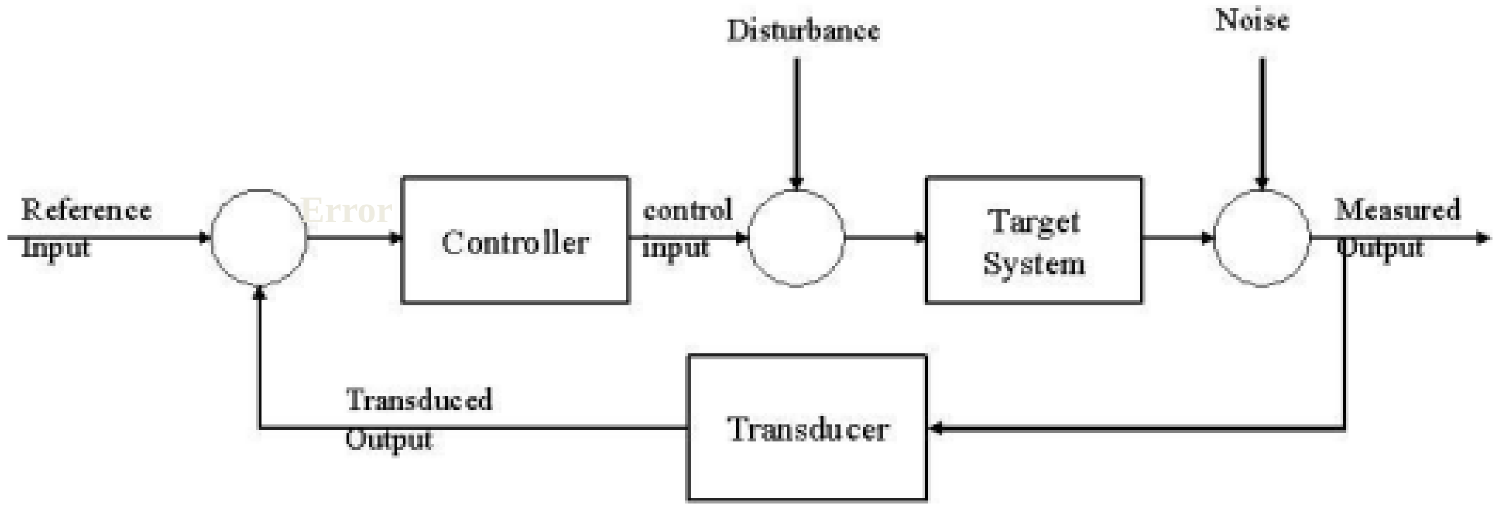
A Feedback Control System


Road surface, gradient
Random variation
Desired Speed
Error
Embedded System that does cruise control
Pedal Angle Setting
Car
Distance Travelled, Time taken
Speed
Speed calculator
Actuator: The entity that *implements* the control
Mechanical Device that Changes pedal angle
Feedback!
Sensor: the ty that asures the output
Odometer, Car clock

A virtual machine as a FBCS


Other Virtual Machines Doing some interfering work
Random variation
Desired Response time
Error
Controller: Process in Domain 0 that periodically changes Cap value of Guest Domain
CPU Cap Value
Xen VM (Guest Domain) With applications running
Response Time (of Applications)
Smoothed Response time
E.g. Smoothing "filter"
Actuator: The entity that *implements* the control
Xen hypervisor scheduler
Sensor: the ty that asures the output
E.g. a "proxy" that Can measure Time taken
Feedback!

A Feedback Control System



<u>Components</u>

- Target system: what is controlled
- Controller: exercises control
- Transducer: translates measured outputs

<u>Data</u>

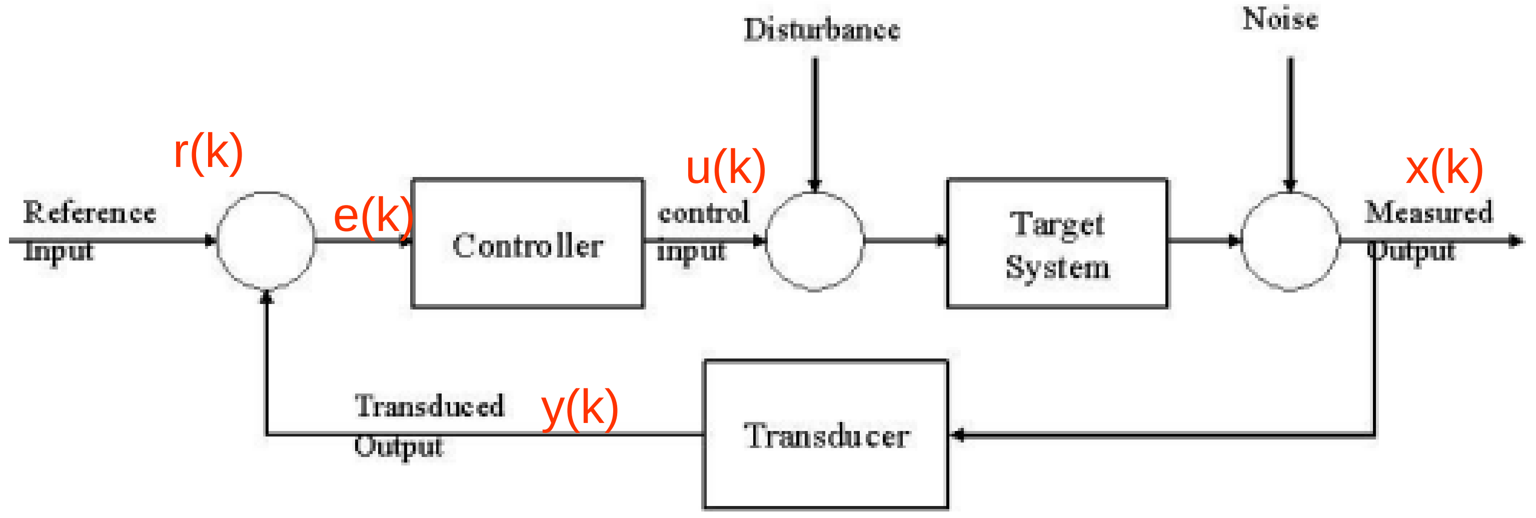
- Reference input: objective
- Control input: manipulated to affect output
- Disturbance input: other factors that affect the target system
- Transduced output: result of manipulation

Given target system, transducer, Control theory finds controller that adjusts control input to achieve given measured output in the presence of disturbances.

Control System Goals

- *Reference Tracking*
  - Ensure that measured output "follows" (tracks) a target desired level
- *Disturbance Rejection*
  - Maintain measured output at a given stable level even in presence of "disturbances"
- *Optimization*
  - No reference input may be given. Maintain measured output and control input at "optimal" levels
    - Minimize petrol consumption, maximize speed (*not available in reality!!)*
    - Minimize power consumed by CPU, maximize performance

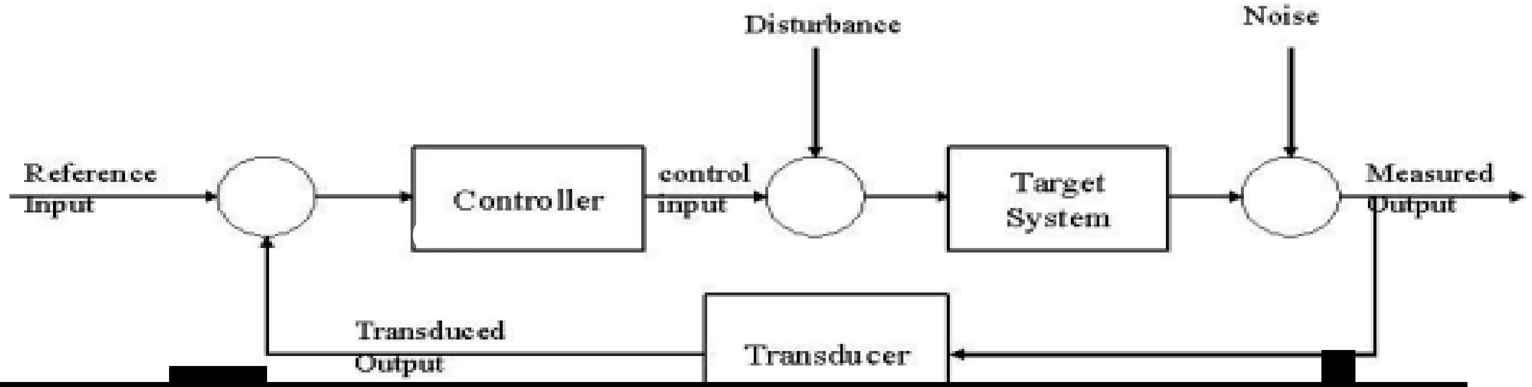
Control System: Basic Working



- "Control Loop" executed every *T* time units.
  - *y(k):* Value of measure at time instant *k*
  - *u(k):* Value of control input at time instant *k*
  - *r(k):* Value of "reference" at time instant *k*
  - *e(k):* Error between reference and measured value
- Next value of control input, i.e. *u(k+1)* to be computed based on *feedback:* e(k)

Determining Change in Control Input

- Change in *u* should depend on relationship between *y* and *u*
- Relationship should be known
  - If known, then why feedback? Why not:
    - y = f(u)
      - (Car speed as function of accelerator pedal angle)
    - → u = f^{-1}(y). For a desired reference, just calculate the control input required by using inverse. This is called *feed forward*



[redacted]

Feedforward (model based)

- Problems:
  - Need accurate model
    - If model wrong, control input can be totally wrong
      - Imagine wrong model b/w accel. pedal and car speed
  - Does not take into account time-dependent behaviour
    - E.g. how long will system require to attain desired value
      - If car was at speed S1, we set cruise control to speed S2, if we set pedal angle to f^{-1}(S2), will it immediately go to speed S2?
  - Cannot take care of “disturbances”
    - E.g. if environmental conditions change, model may not be applicable.
      - What if car was climbing a slope? (And angle was measured on flat ground?)

Feedback addresses many of these problems.

Feedback Control

- Rather than use *only* some offline model, also take into account the immediate measured effect of the value of your control variable
  - If car at S1, want to go to S2, S1 < S2 (positive error), pedal must be pushed further down.
- We still need some idea of the relationship between “input” and “output”
  - Pedal should be pushed? Or released? If pushed – by what angle?
    - (Intuitively, for larger S2 - S1, angle of pushing in must be larger)
  - Another e.g. if e(k) = r(k) – y(k) is positive
    - Should u(k) increase or decrease?
      - If *u* is CPU frequency, *y* is response time. If e(k) is positive, *u* should?
    - Increase/Decrease by how much?
- If increase/decrease too much:
  - Measured output will keep on oscillating above and below reference: Unstable behavior

…Feedback control

- Since we want control to work in a timely manner (not only in “steady-state”), relationship should be time dependent
- New value of *y: y(k+1)* will generally depend on *y(k)* and *u(k)*
  - Speed of car at this instant, will depend on what speed was at the beginning of previous interval, and the pedal angle in the previous interval
  - Similarly, queuing delay at this intv’l will depend on queuing delay of previous instant, and CPU frequency setting of previous interval

UTILITY THEORY

Probability

- probability of an event occurring is the relative frequency with which the event occurs
- if α_i = the probability of event i occurring and there are n possible events (states) then
  - 1. $\alpha_i \geq 0$, i = 1…n
  - 2. $\sum_{i=1}^{n} \alpha_i = 1$
- Lottery (X) with prizes (outcomes, states, events)
- $X_1, X_2, X_3, \ldots, X_n$ with corresponding probabilities
- $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$, respectively (mutually exclusive and exhaustive)
- then the expected value of this lottery is
- $E(X) = \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 + \ldots + \alpha_n X_n$
- $E(X) = \sum_{i=1}^{n} \alpha_i X_i$

Example 1

- Gamble (X) flip of a coin
- if heads, you receive \$1 $X_1 = +1$
- if tails, you pay \$1 $X_2 = -1$
- $E(X) = (0.5)(1) + (0.5)(-1) = 0$

- if you play this game many times, it is likely that you break-even

The Principle of Utility

1. Recognizes the fundamental role of Pain and Pleasure in human life.

2. Approves or disapproves of an action on the basis of the amount of pain or pleasure brought about ("consequences").

3. Equates the good with the pleasurable and evil with pain.

4. Asserts that pleasure and pain are capable of "quantification" -- and hence of measure.

Utility: Some Definitions

- The desirability of preference that individuals or societies have for a given outcome
- Quantitative measure of the attractiveness of a potential outcome

Exposition of a New Theory on the Measurement of Risk (1954)

- Ever since mathematicians first began to study the measurement of risk, there has been general agreement on the following proposition:
- Expected values are computed by multiplying each possible gain by the number of ways in which it can occur, and then dividing the sum of these products by the total number of possible cases.”
- No characteristic of the persons themselves has been taken into consideration.

Why Utility

- Example: buying a lottery ticket
- All men cannot use the same rule to evaluate the gamble. There is no reason to assume that two persons encountering identical risks.
- “The determination of the value of an item must not be based on its price, but rather on the particular circumstances of the person making the estimate.”
- “It becomes evident that no valid measurement of the value of a risk can be obtained without consideration being given to its utility”

Utility and Utility Function

- The utility is dependent on the particular circumstances of the person making the estimate.

$$\Delta U = k\frac{\Delta X}{X}$$

U is utility

K is a constant of proportionality

X is the base amount of wealth

- U(X) = a+b*X
- U(X) = sqrt(X)
- U(X) = log(X)

Decision Theory

- Quantify preferences on outcomes *s*
  - U(*s*,*a*)
- Quantify Beliefs about outcomes of actions
  - *P(s|O,A)* where
    - *O* are observations
    - *A* are actions
- Decision making principle:
  - Choose *A* that Maximizes Expected Utility
    - Needs link between s & A, *s'* = T(*s*,*A*)

Utility

Utility is:

- A numerical measure of goal achievement.
- A numerical measure of ``good''.

Utility isn't just:

- Pleasure.
- Money.
- Happiness.
- Outcomes that have ``usefulness'' for something else.

Utility Matrix

$$U(a,s)$$

OUTCOMES

ACTIONS

	Columns are different things		
Option	1	2	3
A	$u_A(1)$	$u_A(2)$	$u_A(3)$
B	$u_B(1)$	$u_B(2)$	$u_B(3)$

Can we boil all good down to a number?

- Probably not.
- Different kinds of utility (Kahneman):
  - Experienced utility
    - E.g. Pain during treatment
  - Remembered utility
    - E.g. Pain remembered after treatment
  - Predicted utility
    - *Do people know what will be good for them?*
  - Decision utility
    - *Do people use their knowledge when making decisions?*

Fundamental Equation

Value of a decision = Expected Utility of making an action A, where the expectation (average) is carried out over the possible outcomes of that action.

$$E_s\left[U(A \mid s, obs)\right] = \sum_s P(s = \mathbf{R}_{\mathbf{esult}}(A) \mid obs, \mathbf{Do}(A)) U(s = \mathbf{R}_{\mathbf{esult}}(A), A)$$

$s = \mathbf{R}_{\mathbf{esult}}(A)$ is the s^{th} possible outcome of action A

$$V = E\left[U[A \mid s, O]\right] = \sum_s \underbrace{P(s \mid O, A)}_{\text{Transition Matrix}} \underbrace{U(s, A)}_{\text{Utility Function}}$$

s : state of the world

O: observation

A : action

Preference Nomenclature

Lotteries:

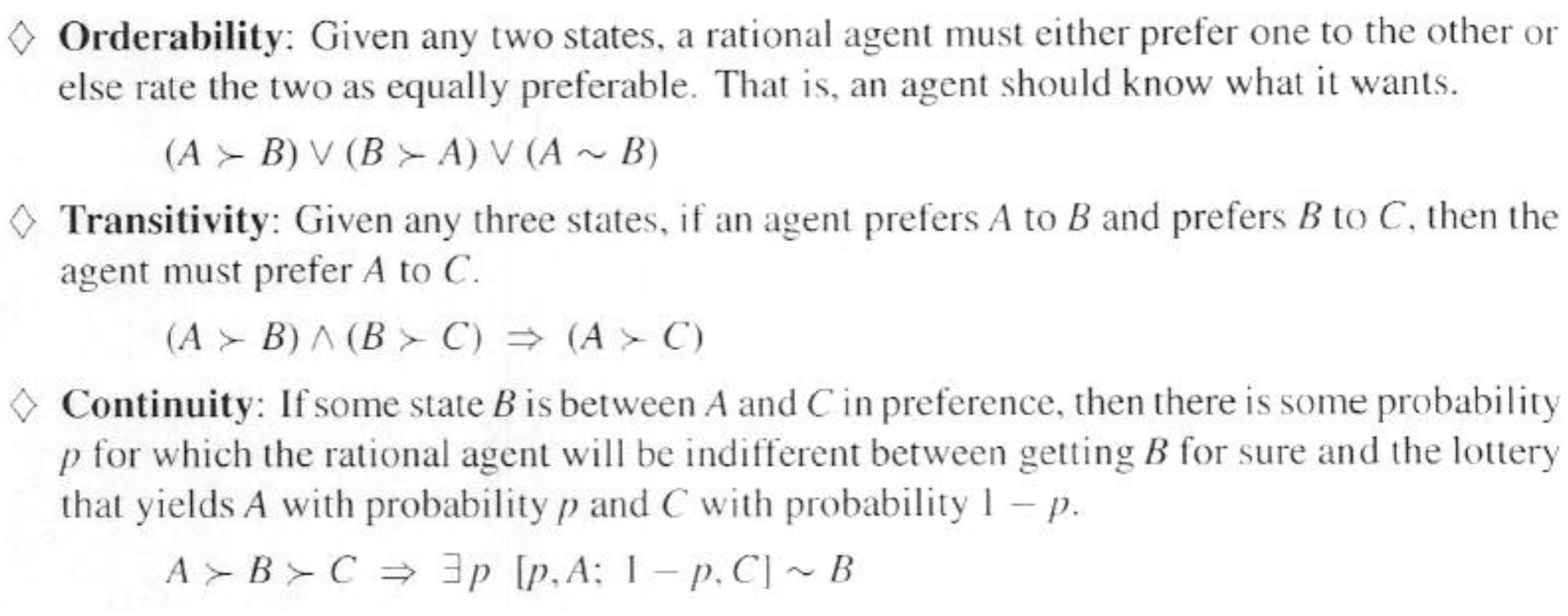
A lottery is a probabilistic mixture of outcomes

$$L = [p, A;\ 1 - p, B]$$

Ordering using lotteries

$A \succ B$	A is preferred to B
$A \sim B$	the agent is indifferent between A and B
$A \succsim B$	the agent prefers A to B or is indifferent between them

Utility Theory Axioms 1



◇ **Orderability**: Given any two states, a rational agent must either prefer one to the other or else rate the two as equally preferable. That is, an agent should know what it wants.

$$(A \succ B) \lor (B \succ A) \lor (A \sim B)$$

◇ **Transitivity**: Given any three states, if an agent prefers A to B and prefers B to C, then the agent must prefer A to C.

$$(A \succ B) \land (B \succ C) \;\Rightarrow\; (A \succ C)$$

◇ **Continuity**: If some state B is between A and C in preference, then there is some probability p for which the rational agent will be indifferent between getting B for sure and the lottery that yields A with probability p and C with probability $1-p$.

$$A \succ B \succ C \;\Rightarrow\; \exists p \;\; [p, A;\; 1-p, C] \sim B$$

Utility Axioms 2

◇ **Substitutability**: If an agent is indifferent between two lotteries, A and B, then the agent is indifferent between two more complex lotteries that are the same except that B is substituted for A in one of them. This holds regardless of the probabilities and the other outcome(s) in the lotteries.

$$A \sim B \;\Rightarrow\; [p, A;\; 1-p, C] \sim [p, B;\; 1-p, C]$$

◇ **Monotonicity**: Suppose there are two lotteries that have the same two outcomes, A and B. If an agent prefers A to B, then the agent must prefer the lottery that has a higher probability for A (and vice versa).

$$A \succ B \;\Rightarrow\; (p \geq q \;\Leftrightarrow\; [p, A;\; 1-p, B] \succsim [q, A;\; 1-q, B])$$

◇ **Decomposability**: Compound lotteries can be reduced to simpler ones using the laws of probability. This has been called the "no fun in gambling" rule because it says that an agent should not prefer (or disprefer) one lottery just because it has more choice points than another.[2]

$$[p, A;\; 1-p, [q, B;\; 1-q, C]] \sim [p, A;\; (1-p)q, B;\; (1-p)(1-q), C]$$

What do the Axioms do? They Guarantee:

1) Utility principle

$$U(A) > U(B) \Leftrightarrow A \succ B$$
$$U(A) = U(B) \Leftrightarrow A \sim B$$

There exists a monotonic function that numerically encodes preferences

2) Maximum expected utility principle

$$U([p_1, S_1; \ldots ; p_n, S_n]) = \sum_i p_i U(S_i)$$

Utility of a lottery is the expection of the utilities

An Example: You bet your *what*?

You just won $1,000,000

An Example: You bet your *what*?

You just won $1,000,000

BUT

You are offered a gamble:

Bet your $1,000,000.00 on a fair coin flip.

Heads: $3,000,000

Tails: $0.00

What should you do?

Problem Analysis

Expected monetary gain = 0.5* \$0 + 0.5* \$ 3,000,000 =
\$1,500,000

\$1,500,000 > \$ 1,000,000 !
Will you take the bet now?
How much do you need as a pay off?

Utility theory posits lotteries that result in indifference, and in taking the bet.

Let S_k be your current wealth.

Let U(S_k) = 5; U($S_{k\ +3,000,000}$) = 10; U($S_{k+1,000,000}$) = 8;

$$EU(Accept) = \tfrac{1}{2}U(S_k) + \tfrac{1}{2}U(S_{k+3,000,000})$$

$$EU(Accept) = U(S_{k+1,000,000})$$

AUCTION THEORY

SOME EXAMPLES

FCC and Radio Spectrum

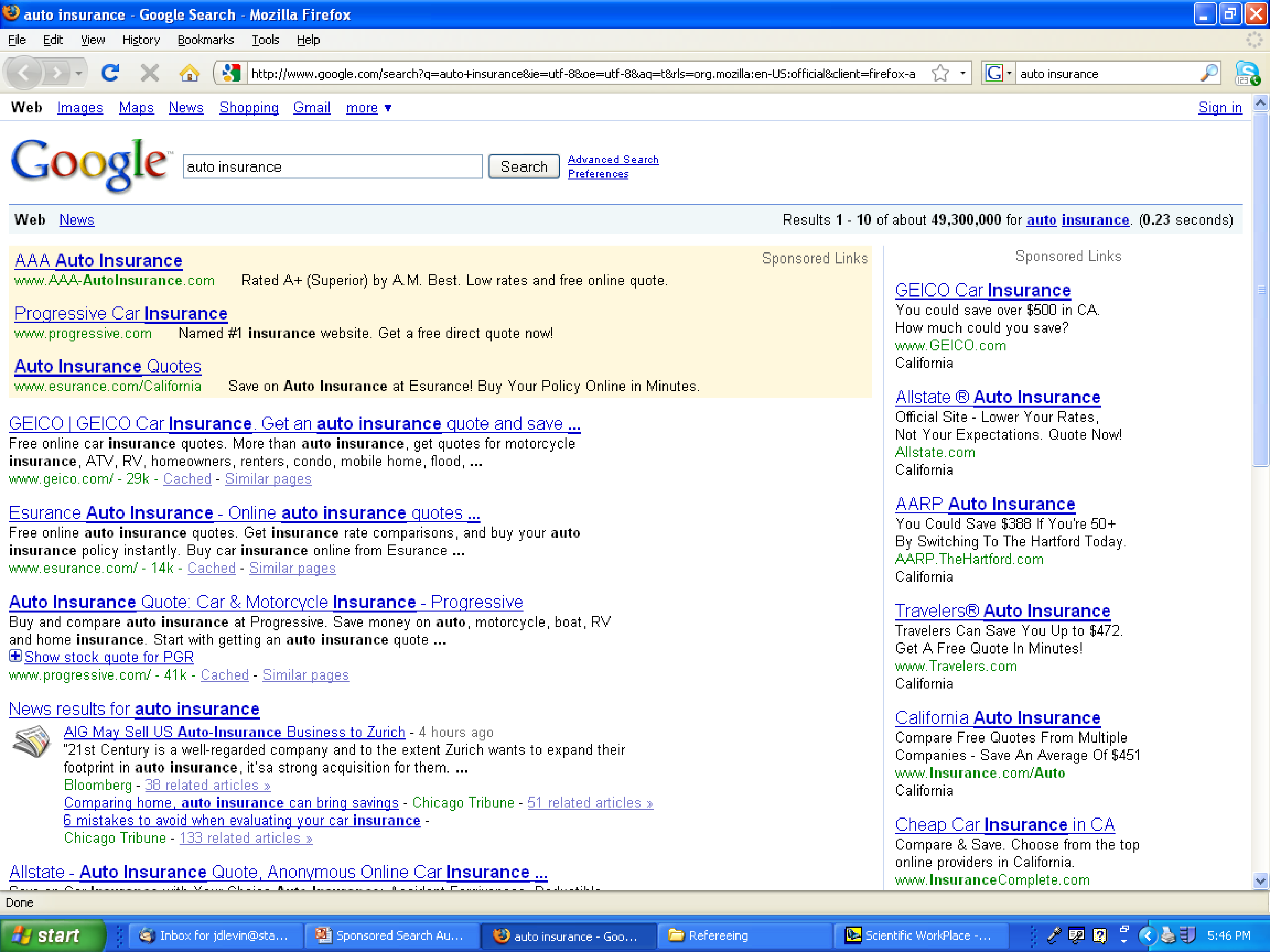
- FCC regulates use of electromagnetic radio spectrum: used for broadcast TV, radio, cell phones, WiFi, etc.
- Why regulate?
  - There is a limited amount of spectrum
  - There are many potential users
  - There are interference problems if users overlap.
- So, how should the FCC decide who gets a license to use spectrum?
- Historically, licenses were allocated administratively (TV & radio stations) or by lottery.

Spectrum auctions

- Coase (1959) suggested that the FCC should auction spectrum licenses.
  - If there were no transaction costs, the initial assignment of ownership wouldn’t matter (the Coase Theorem).
  - But the real world isn’t like that… decentralized trade may not lead to efficient allocations (more on this later).
- In the early 1990s, the FCC started to think about auctions as a way to allocate licenses efficiently, and adopted a new design proposed by Stanford economists.
  - Many countries now use auctions that result in hundreds of millions, or billions, of dollars of government revenue.

Sponsored search auctions

- Google revenue in 2008: **$21,795,550,000**.
- Hal Varian, Google chief economist:
  - "What most people don't realize is that all that money comes pennies at a time."
- Google revenue comes from selling ads: there is an auction each time someone enters a search query
  - Bids in the auction determine the ads that appear on the RHS of the page, and sometimes the top.
  - Google design evolved from earlier, and problematic, design used by Overture (now part of Yahoo!)
- We will see that Google's design is closely related to the theory of matching we've already studied.


auto insurance - Google Search - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.google.com/search?q=auto+insurance&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a

Web Images Maps News Shopping Gmail more ▾ Sign in

auto insurance Search Advanced Search Preferences

Web News Results **1** - **10** of about **49,300,000** for **auto insurance**. (**0.23** seconds)

Sponsored Links

AAA **Auto Insurance**
www.AAA-**AutoInsurance**.com Rated A+ (Superior) by A.M. Best. Low rates and free online quote.

Progressive Car **Insurance**
www.progressive.com Named #1 **insurance** website. Get a free direct quote now!

Auto Insurance Quotes
www.esurance.com/California Save on **Auto Insurance** at Esurance! Buy Your Policy Online in Minutes.

GEICO | GEICO Car **Insurance**. Get an **auto insurance** quote and save ...
Free online car **insurance** quotes. More than **auto insurance**, get quotes for motorcycle **insurance**, ATV, RV, homeowners, renters, condo, mobile home, flood, ...
www.geico.com/ - 29k - Cached - Similar pages

Esurance **Auto Insurance** - Online **auto insurance** quotes ...
Free online **auto insurance** quotes. Get **insurance** rate comparisons, and buy your **auto insurance** policy instantly. Buy car **insurance** online from Esurance ...
www.esurance.com/ - 14k - Cached - Similar pages

Auto Insurance Quote: Car & Motorcycle **Insurance** - Progressive
Buy and compare **auto insurance** at Progressive. Save money on **auto**, motorcycle, boat, RV and home **insurance**. Start with getting an **auto insurance** quote ...
Show stock quote for PGR
www.progressive.com/ - 41k - Cached - Similar pages

News results for **auto insurance**
AIG May Sell US **Auto-Insurance** Business to Zurich - 4 hours ago
"21st Century is a well-regarded company and to the extent Zurich wants to expand their footprint in **auto insurance**, it'sa strong acquisition for them. ...
Bloomberg - 38 related articles »
Comparing home, **auto insurance** can bring savings - Chicago Tribune - 51 related articles »
6 mistakes to avoid when evaluating your car **insurance** -
Chicago Tribune - 133 related articles »

Allstate - **Auto Insurance** Quote, Anonymous Online Car **Insurance** ...

Sponsored Links

GEICO Car **Insurance**
You could save over $500 in CA.
How much could you save?
www.GEICO.com
California

Allstate ® **Auto Insurance**
Official Site - Lower Your Rates,
Not Your Expectations. Quote Now!
Allstate.com
California

AARP **Auto Insurance**
You Could Save $388 If You're 50+
By Switching To The Hartford Today.
AARP.TheHartford.com
California

Travelers® **Auto Insurance**
Travelers Can Save You Up to $472.
Get A Free Quote In Minutes!
www.Travelers.com
California

California **Auto Insurance**
Compare Free Quotes From Multiple
Companies - Save An Average Of $451
www.**Insurance**.com/**Auto**
California

Cheap Car **Insurance** in CA
Compare & Save. Choose from the top
online providers in California.
www.**Insurance**Complete.com

Done

start | Inbox for jdlevin@sta... | Sponsored Search Au... | auto insurance - Goo... | Refereeing | Scientific WorkPlace -... | 5:46 PM

Auctions everywhere…

- Auctions are commonly used to sell (and buy) goods that are idiosyncratic or hard to price.
  - Real estate
  - Art, antiques, estates
  - Collectibles (eBay)
  - Used cars, equipment
  - Emissions permits
  - Natural resources: timber, gas, oil, radio spectrum…
  - Financial assets: treasury bills, corporate debt.
  - Bankruptcy auctions
  - Sale of companies: privatization, IPOs, take-overs, etc.
  - Procurement: highways, construction, defense.
- Auction theory also has close and beautiful ties to standard price theory (monopoly theory) and matching.

Auction Theory

Selling a single good

- We’ll start with sale of a single good.
- Why not just set a price?
  - Seller may not know what price to set.
  - Potential buyers know what they’d pay, but aren’t telling.
  - Remember from last time – auction serves as a mechanism for *price discovery.*
- Let’s consider different ways to run an auction.

Canonical model

- Potential buyers
  - Two bidders (later *N* bidders)
  - Each bidder *i* has value v_i
  - Each v_i drawn from uniform distribution on [0,1].
- Same ideas will apply with *N* bidders and a value distribution other than U[0,1].
- Seller gets to set the auction rules.

Ascending auction

- Price starts at zero, and rises slowly.
- Buyers indicate their willingness to continue bidding (e.g. keep their hand up) or can exit.
- Auction ends when just one bidder remains.
- Final bidder wins, and pays the price at which the second remaining bidder dropped out.

- How should you bid?

Ascending Auction

- Optimal strategy: continue bidding until the price just equals your value.
- Bidder with highest value will win.
- Winner will pay second highest value.

- Example with three bidders
  - Suppose values are 25, 33, and 75.
  - First exits at 25, second at 33 and auction ends.

Properties of Auctions

- Allocative efficiency means that in all these auctions the highest bidder always wins (i.e., there are no reserve prices).
- It is desirable for an auction to be computationally efficient.
- Revenue Equivalence Theorem: Any two auctions such that:
  - *The bidder with the highest value wins*
  - *The bidder with the lowest value expects zero profit*
  - *Bidders are risk-neutral 1*
  - *Value distributions are strictly increasing and atomless*

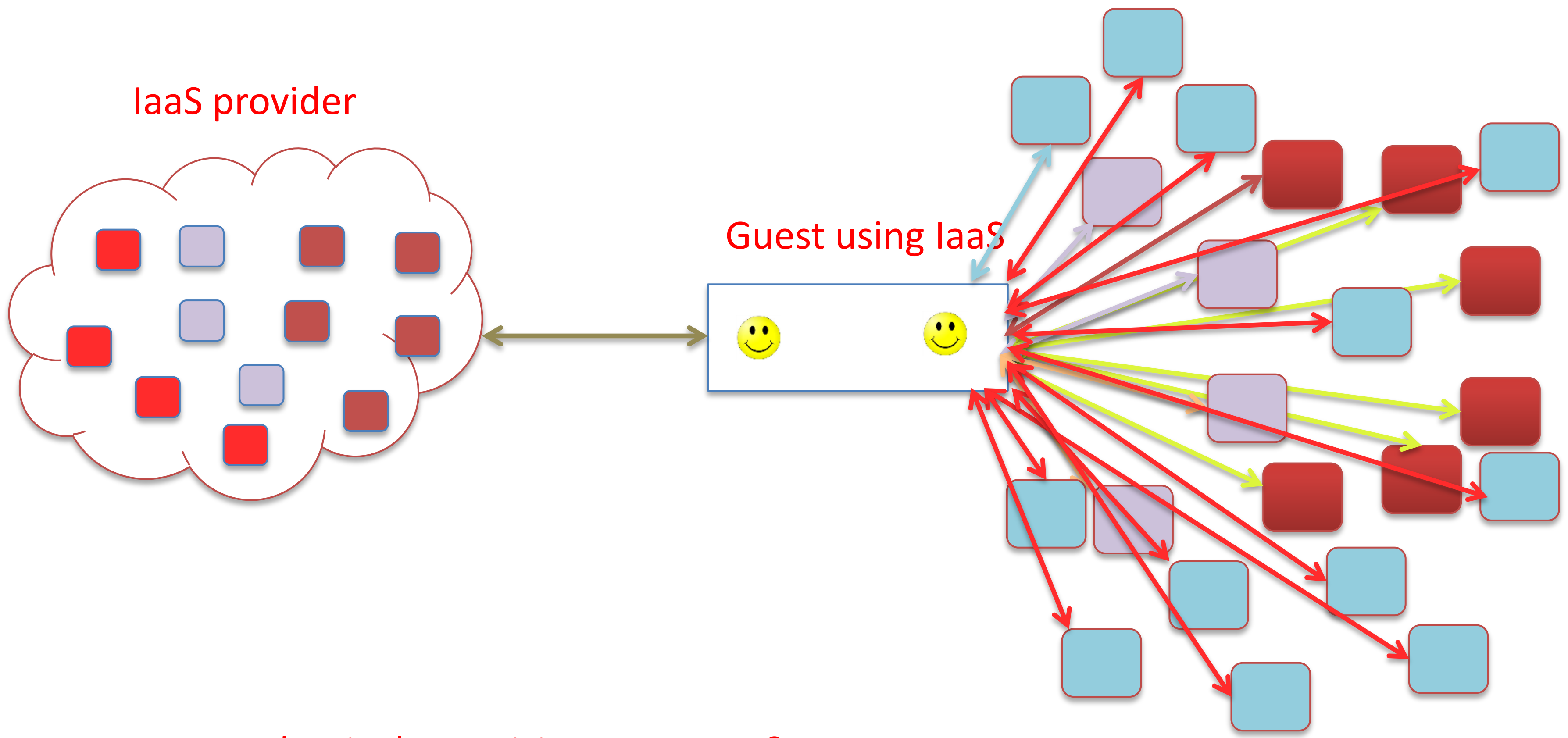
  have the same revenue and also the same expected profit for each bidder. The theorem can help find some equilibrium strategy.

Summary on Auction Theory

- Auctions take many forms but always satisfy two conditions:
  - They may be used to sell any item and so are *universal*, also
  - The outcome of the auction does not depend on the identity of the bidders; i.e., auctions are *anonymous*.
- Properties
  - efficiency of a given auction design
  - optimal and equilibrium bidding strategies
  - revenue comparison.
- Winner curse
  - winner will frequently have bid too much for the auctioned item.

PROPORTIONAL THRESHOLDING

Problem



How to adaptively provision resources?

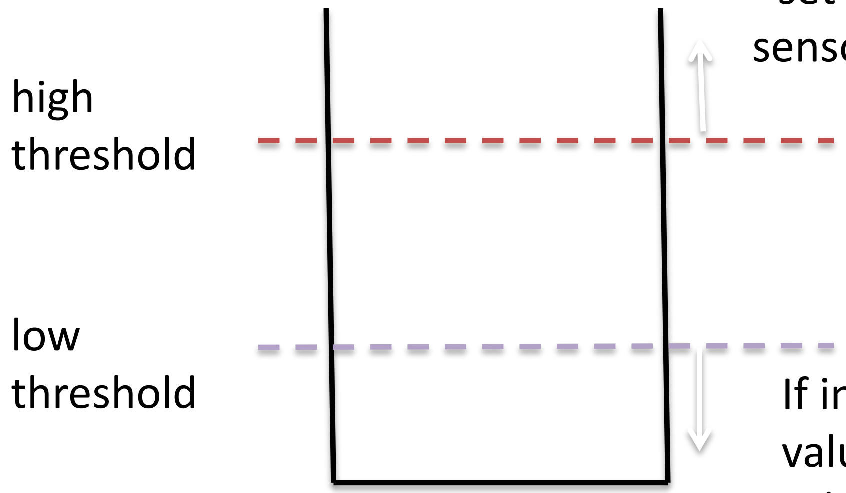
Challenges

- Decoupling control
  - Cloud controller
    - arbitrate resource requests, select guest VM placements
  - Application controller
    - determine physical resources needed and communicate to cloud controller
- Control granularity
  - Coarse sensor and actuator information.
  - Noisy sensor measurement
    - CPU utilization as percentage of VM usage
    - work-conserving scheduler gives noisy measurement

Proposed solution

- A feedback driven application control implemented at the guest's end.
- Guest application controllers or *slice controllers.*
- IaaS provider provides sensors and actuators to enable control policies.
- Slice controllers use APIs to collect coarse-grained information from sensors and actuators.
- Solution: A control technique, *proportional thresholding,* for coarse-grained actuators with a wide range of actuator values.

Proportional thresholding



If incoming accumulated sensor value > high threshold,
- then request resources
- set high threshold to accumulated sensor value

If incoming accumulated sensor value < low threshold,
- then release resources
- set low threshold to accumulated sensor value

Why proportional thresholding?

- Parameters to tune: CPU entitlement and utilization
- Tuned using: an integral control
  - control effort is proportional to the integral of the error
  - well-suited for coarse-grained actuators
  - actuators have a dynamic target range
  - steady state error is zero

RESOURCE MANAGEMENT AND SCHEDULING

Resource management and scheduling

- Critical function of any man-made system.
- It affects the three basic criteria for the evaluation of a system:
  - Functionality.
  - Performance.
  - Cost.
- Scheduling in a computing system → deciding how to allocate resources of a system, such as CPU cycles, memory, secondary storage space, I/O and network bandwidth, between users and tasks.
- Policies and mechanisms for resource allocation.
  - Policy → principles guiding decisions.
  - Mechanisms → the means to implement policies.

Implementing Policies

- Implementation literally means carrying out, accomplishing, fulfilling, producing or completing a given task.
- Policy implementation encompasses those actions by individuals or groups that are directed at the achievement of objectives set forth in policy decisions.
  - one-time efforts: to transform decisions into operational terms and
  - continuing efforts: to achieve the large and small changes mandated by policy decisions

Cloud Resource Management

- Cloud resource management
  - Requires complex policies and decisions for multi-objective optimization.
  - It is challenging - the complexity of the system makes it impossible to have accurate global state information.
  - Affected by unpredictable interactions with the environment, e.g., system failures, attacks.
  - Cloud service providers are faced with large fluctuating loads which challenge the claim of cloud elasticity.
- The strategies for resource management for IaaS, PaaS, and SaaS are different.

Cloud resource management (CRM) policies

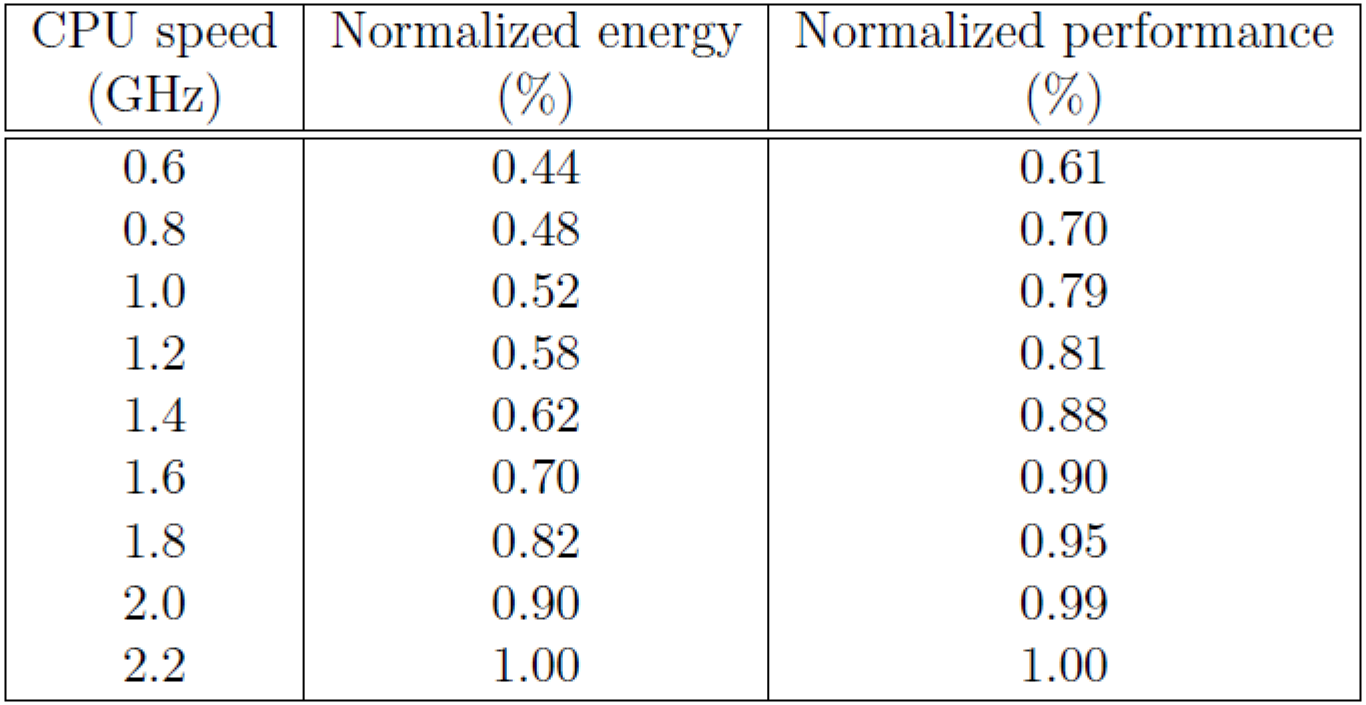
- Admission control → prevent the system from accepting workload in violation of high-level system policies.
- Capacity allocation → allocate resources for individual activations of a service.
- Load balancing → distribute the workload evenly among the servers.
- Energy optimization → minimization of energy consumption.
- Quality of service (QoS) guarantees → ability to satisfy timing or other conditions specified by a Service Level Agreement.

Mechanisms for the implementation of resource management policies

- Control theory → uses the feedback to guarantee system stability and predict transient behavior.
- Machine learning → does not need a performance model of the system.
- Utility-based → require a performance model and a mechanism to correlate user-level performance with cost.
- Market-oriented/economic → do not require a model of the system, e.g., combinatorial auctions for bundles of resources.

Tradeoffs

- To reduce cost and save energy we may need to concentrate the load on fewer servers rather than balance the load among them.
- We may also need to operate at a lower clock rate; the performance decreases at a lower rate than does the energy.



CPU speed (GHz)	Normalized energy (%)	Normalized performance (%)
0.6	0.44	0.61
0.8	0.48	0.70
1.0	0.52	0.79
1.2	0.58	0.81
1.4	0.62	0.88
1.6	0.70	0.90
1.8	0.82	0.95
2.0	0.90	0.99
2.2	1.00	1.00

Control theory application to cloud resource management (CRM)

- The main components of a control system:
  - The inputs:
    - the offered workload and the policies for admission control, the capacity allocation, the load balancing, the energy optimization, and the QoS guarantees in the cloud.
  - The control system components:
    - *sensors* used to estimate relevant measures of performance and *controllers* which implement various policies.
  - The outputs:
    - the resource allocations to the individual applications.

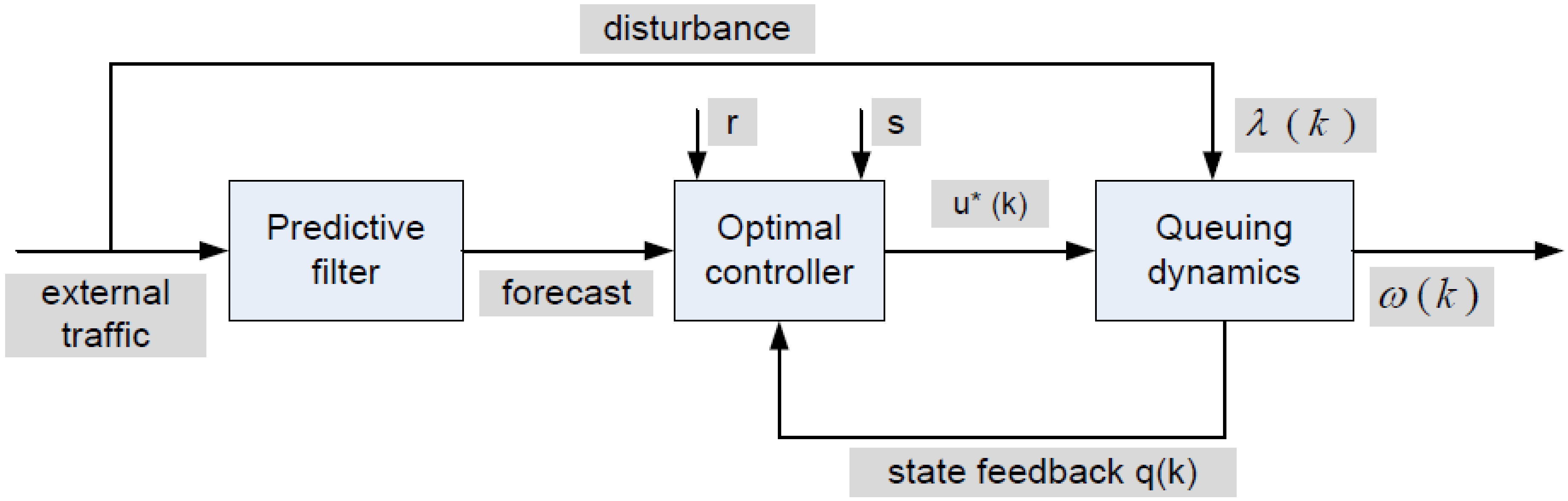
Feedback and Stability

- Control granularity: the level of detail of the information used to control the system.
  - Fine control → very detailed information about the parameters controlling the system state is used.
  - Coarse control → the accuracy of these parameters is traded for the efficiency of implementation.
- The controllers use the feedback provided by sensors to stabilize the system.
  - Stability is related to the change of the output.

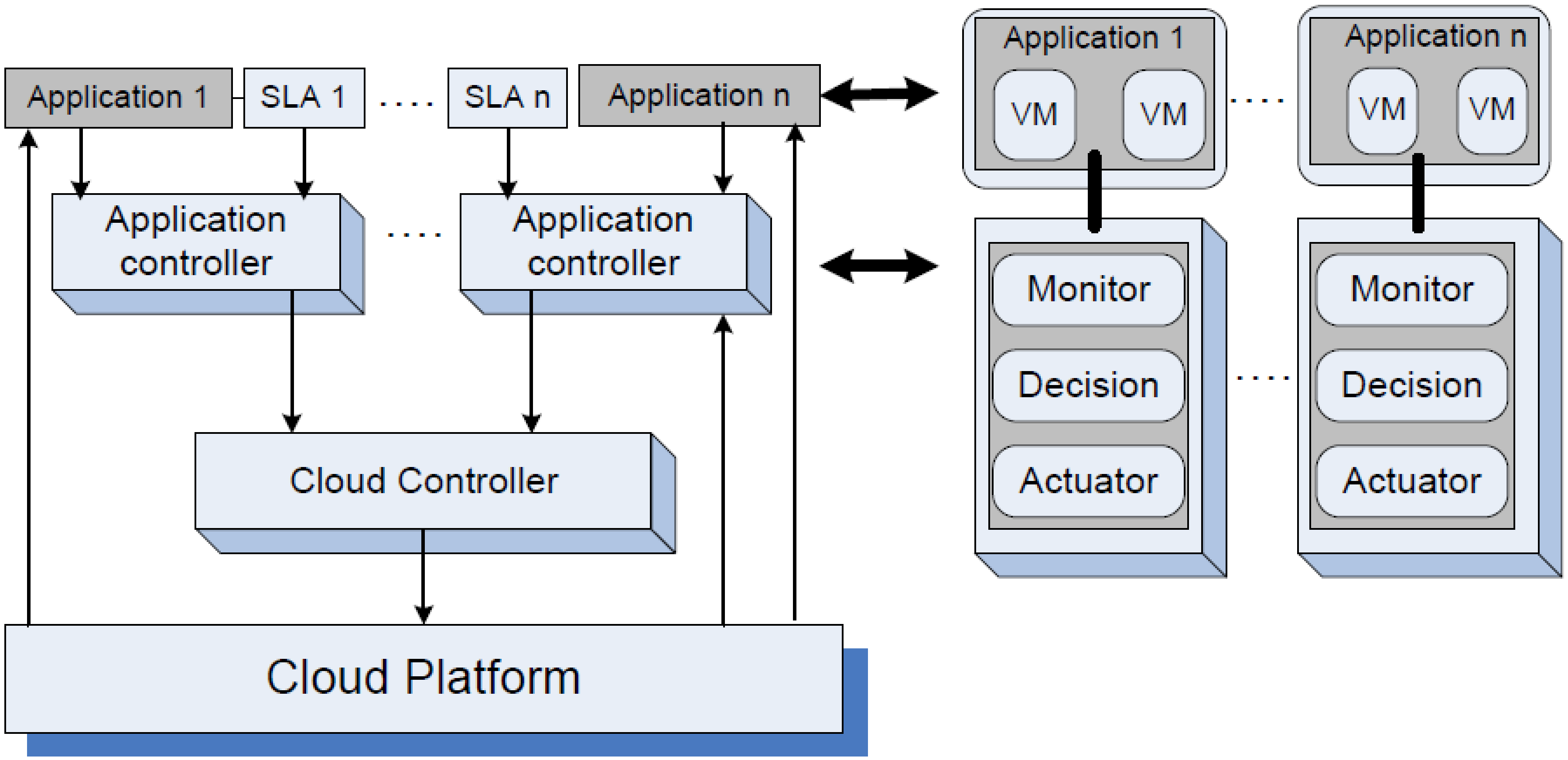
Feedback and Stability

- Sources of instability in any control system:
  - The delay in getting the system reaction after a control action.
  - The granularity of the control, the fact that a small change enacted by the controllers leads to very large changes of the output.
  - Oscillations, when the changes of the input are too large and the control is too weak, such that the changes of the input propagate directly to the output.

Cloud Controller



The controller uses the feedback regarding the current state and the estimation of the future disturbance due to environment to compute the optimal inputs over a finite horizon. ***r*** and ***s*** are the weighting factors of the performance index.


Application 1
SLA 1
SLA n
Application n
Application controller
Application controller
Cloud Controller
Cloud Platform
Application 1
VM
VM
Application n
VM
VM
Monitor
Decision
Actuator
Monitor
Decision
Actuator

Lessons from the two-level experiment

- The actions of the control system should be carried out in a rhythm that does not lead to instability.
- Adjustments should only be carried out after the performance of the system has stabilized.
- If upper and a lower thresholds are set, then instability occurs when they are too close to one another if the variations of the workload are large enough and the time required to adapt does not allow the system to stabilize.
- The actions consist of allocation/deallocation of one or more virtual machines. Sometimes allocation/dealocation of a single VM required by one of the threshold may cause crossing of the other, another source of instability.

Control theory application to CRM

- Regulate the key operating parameters of the system based on measurement of the system output.
- The feedback control assumes a linear time-invariant system model, and a closed-loop controller.
- The system transfer function satisfies stability and sensitivity constraints.
- A threshold → the value of a parameter related to the state of a system that triggers a change in the system behavior.
- Thresholds → used to keep critical parameters of a system in a predefined range.
- Two types of policies:
  - 1. threshold-based → upper and lower bounds on performance trigger adaptation through resource reallocation; such policies are simple and intuitive but require setting per-application thresholds.
  - 2. sequential decision → based on Markovian decision models.

Design decisions

- Is it beneficial to have two types of controllers:
- Application controllers determine if additional resources are needed.
- Cloud controllers arbitrate requests for resources and allocates the physical resources.
- Choose fine versus coarse control.
- Dynamic thresholds based on time averages better versus static ones.
- Use a high and a low threshold versus a high threshold only.

FEEDBACK CONTROL BASED ON DYNAMIC THRESHOLDS

Proportional thresholding

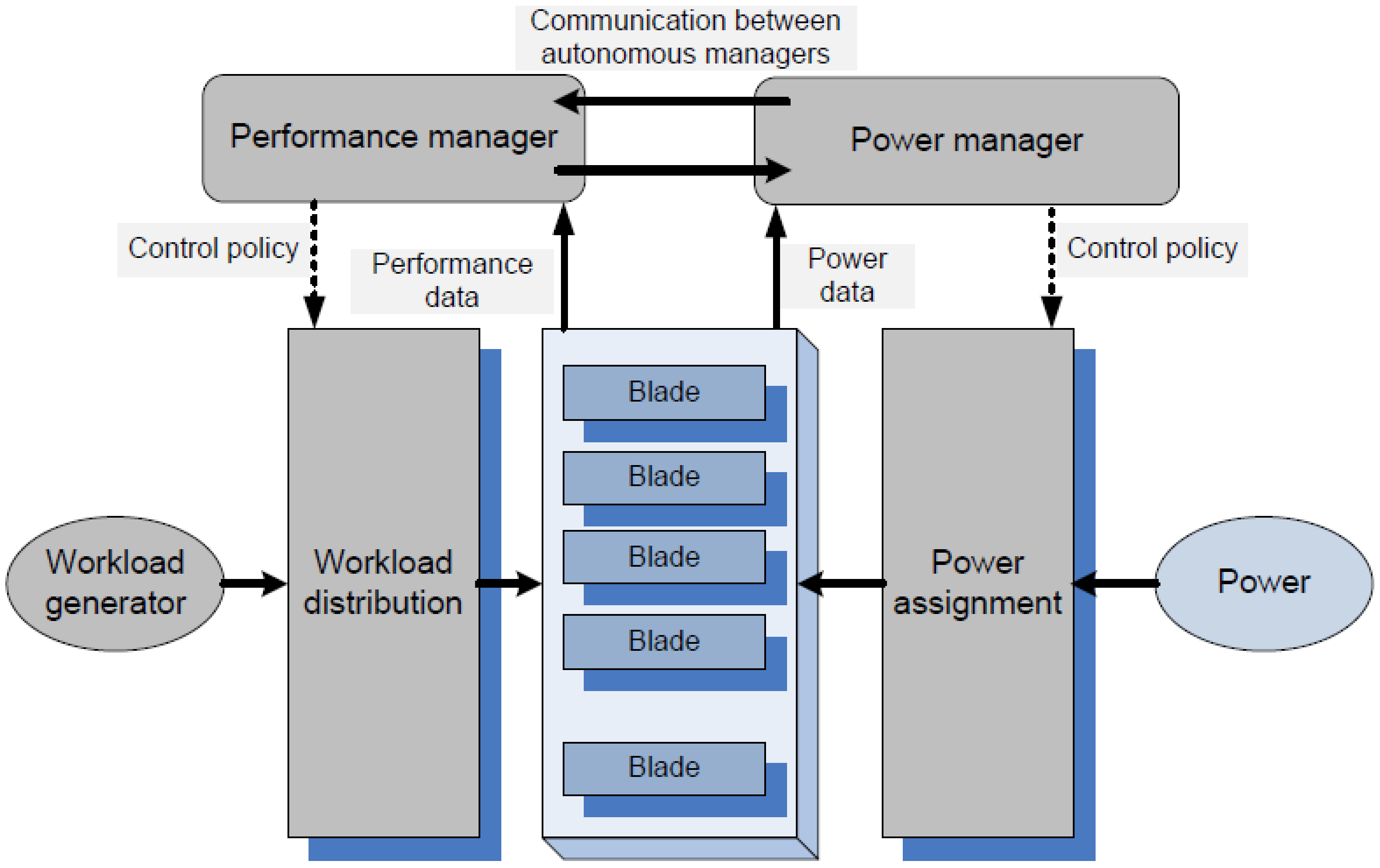
- Algorithm
  - Compute the integral value of the high and the low threshold as averages of the maximum and, respectively, the minimum of the processor utilization over the process history.
  - Request additional VMs when the average value of the CPU utilization over the current time slice exceeds the high threshold.
  - Release a VM when the average value of the CPU utilization over the current time slice falls below the low threshold.
- Conclusions
  - Dynamic thresholds perform better than the static ones.
  - Two thresholds are better than one.

COORDINATION OF SPECIALIZED AUTONOMIC PERFORMANCE MANAGERS

Coordinating power and performance management

- Use separate controllers/managers for the two objectives.
- Identify a minimal set of parameters to be exchanged between the two managers.
- Use a joint utility function for power and performance.
- Set up a power cap for individual systems based on the utility-optimized power management policy.
- Use a standard performance manager modified only to accept input from the power manager regarding the frequency determined according to the power management policy.
- Use standard software systems.

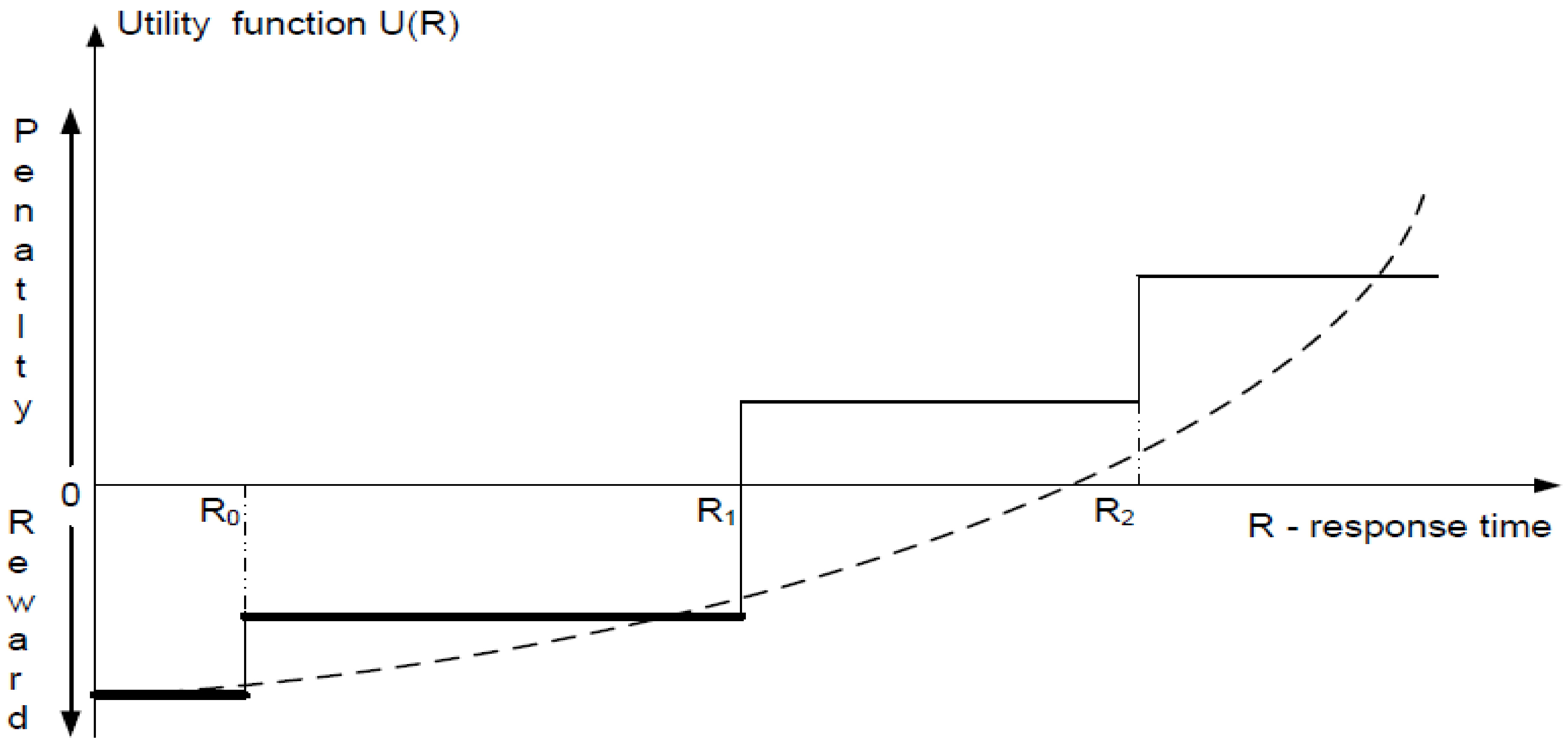
Autonomous performance and power managers cooperate to ensure prescribed performance and energy optimization; they are fed with performance and power data and implement the performance and power management policies



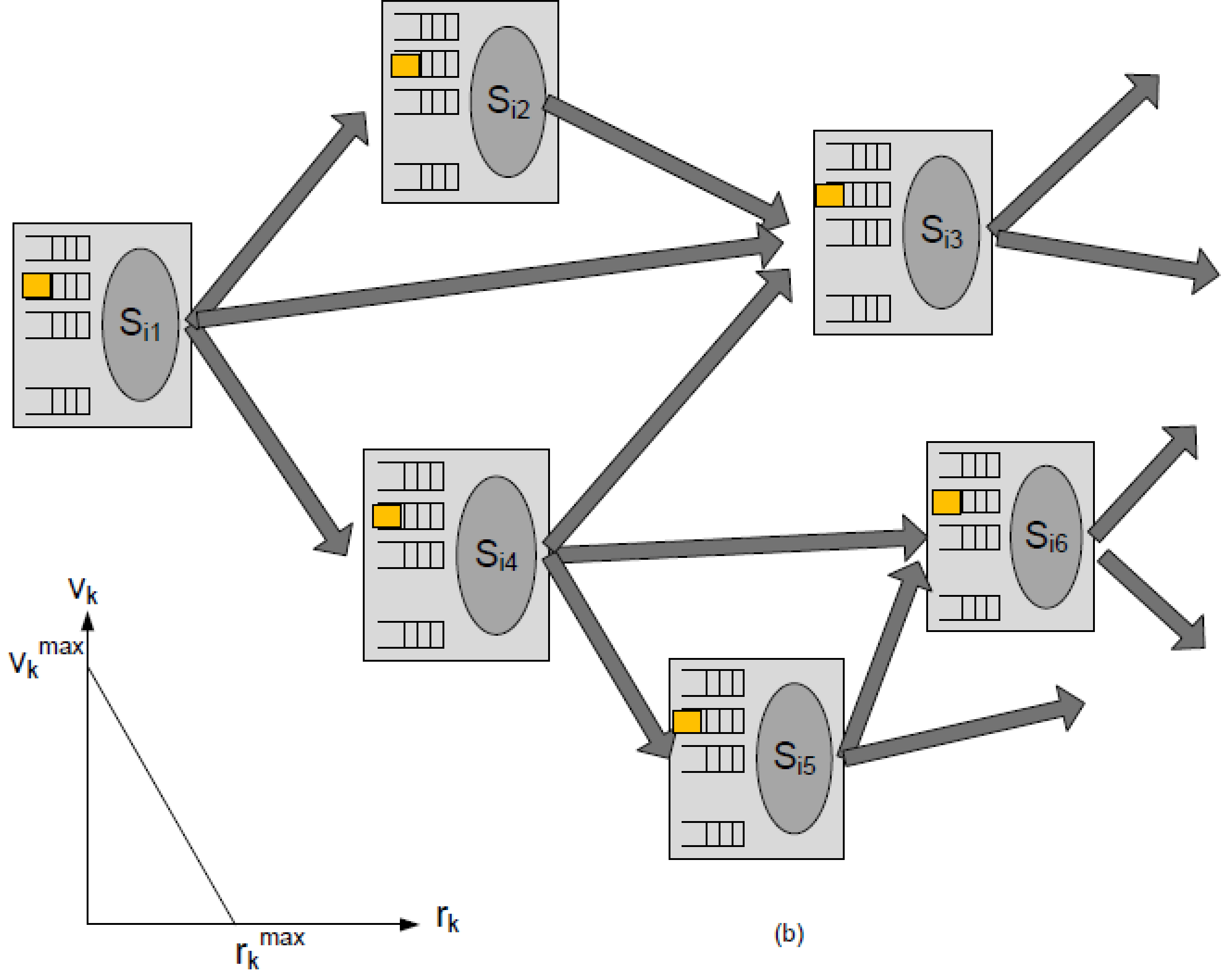
A UTILITY-BASED MODEL FOR CLOUD-BASED WEB SERVICES

A utility-based model for cloud-based web services

- A service level agreement (SLA) → specifies the rewards as well as penalties associated with specific performance metrics.
- The SLA for cloud-based web services uses the average response time to reflect the Quality of Service.
- We assume a cloud providing *K* different classes of service, each class *k* involving *Nk* applications.
- The system is modeled as a network of queues with multi-queues for each server.
- A delay center models the think time of the user after the completion of service at one server and the start of processing at the next server.



The utility function U(R) is a series of step functions with jumps corresponding to the response time, R=R0 | R1 | R2, when the reward and the penalty levels change according to the SLA. The dotted line shows a quadratic approximation of the utility function.



(a)

(a) The utility function: *vk* the revenue (or the penalty) function of the response time *rk* for a request of class *k*.
(b) A network of multiqueues.

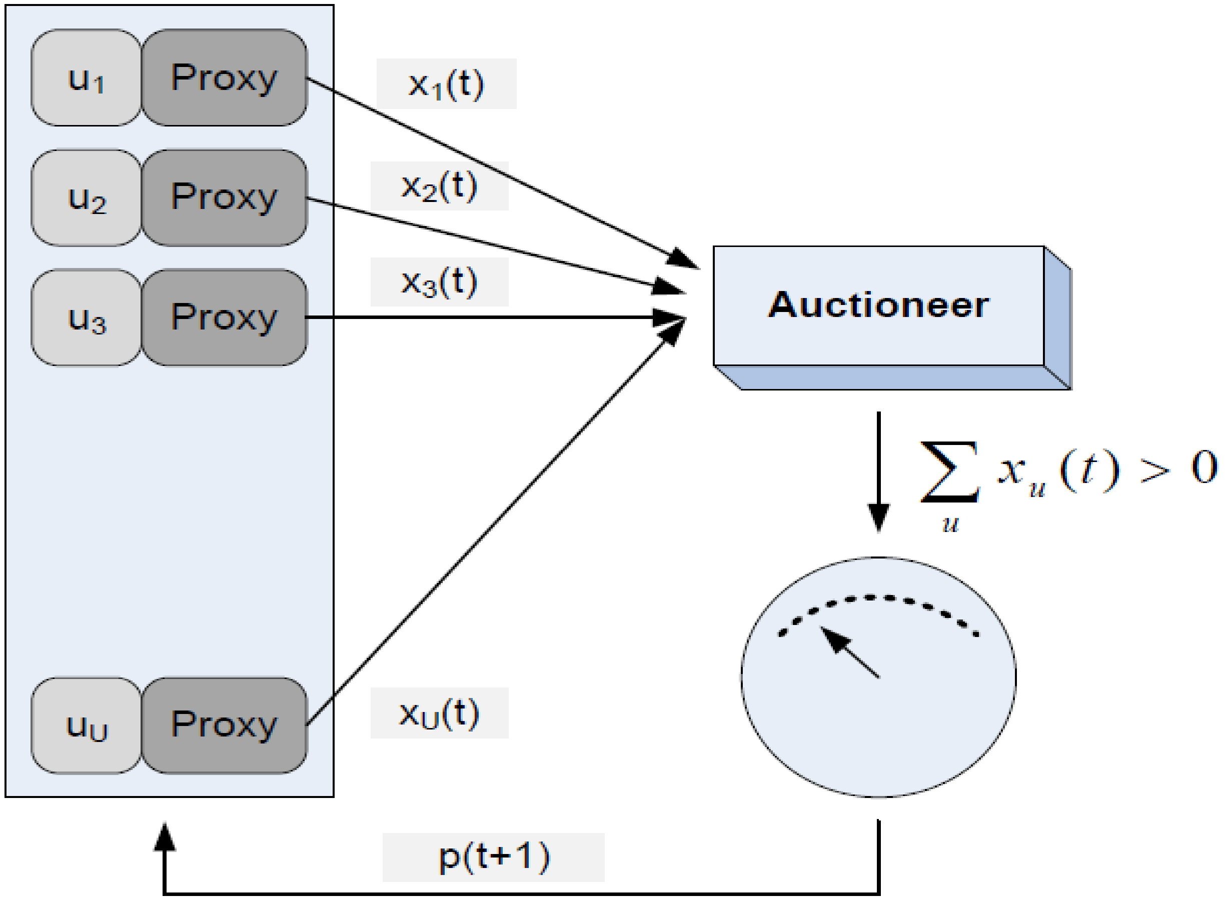
RESOURCE BUNDLING; COMBINATORIAL AUCTIONS FOR CLOUD RESOURCES

Resource bundling

- Resources in a cloud are allocated in bundles.
- Users get maximum benefit from a specific combination of resources: CPU cycles, main memory, disk space, network bandwidth, and so on.
- Resource bundling complicates traditional resource allocation models and has generated an interest in economic models and, in particular, in auction algorithms.
- The bidding process aims to optimize an objective function *f(x,p)*.
- In the context of cloud computing, an auction is the allocation of resources to the highest bidder.

Combinatorial auctions for cloud resources

- Users provide bids for desirable bundles and the price they are willing to pay.
- Prices and allocation are set as a result of an auction.
- Ascending Clock Auction, (ASCA) → the current price for each resource is represented by a "clock" seen by all participants at the auction.
- The algorithm involves user bidding in multiple rounds; to address this problem the user proxies automatically adjust their demands on behalf of the actual bidders.



The schematics of the ASCA algorithm; to allow for a single round auction users are represented by proxies which place the bids *xu(t)*. The auctioneer determines if there is an excess demand and, in that case, it raises the price of resources for which the demand exceeds the supply and requests new bids.

Pricing and allocation algorithms

- A pricing and allocation algorithm partitions the set of users in two disjoint sets, winners and losers.
- Desirable properties of a pricing algorithm:
  - Be computationally tractable; traditional combinatorial auction algorithms e.g., Vickey-Clarke-Groves (VLG) are not computationally tractable.
  - Scale well - given the scale of the system and the number of requests for service, scalability is a necessary condition.
  - Be objective - partitioning in winners and losers should only be based on the price of a user's bid; if the price exceeds the threshold then the user is a winner, otherwise the user is a loser.
  - Be fair - make sure that the prices are uniform, all winners within a given resource pool pay the same price.
  - Indicate clearly at the end of the auction the unit prices for each resource pool.
  - Indicate clearly to all participants the relationship between the supply and the demand in the system.

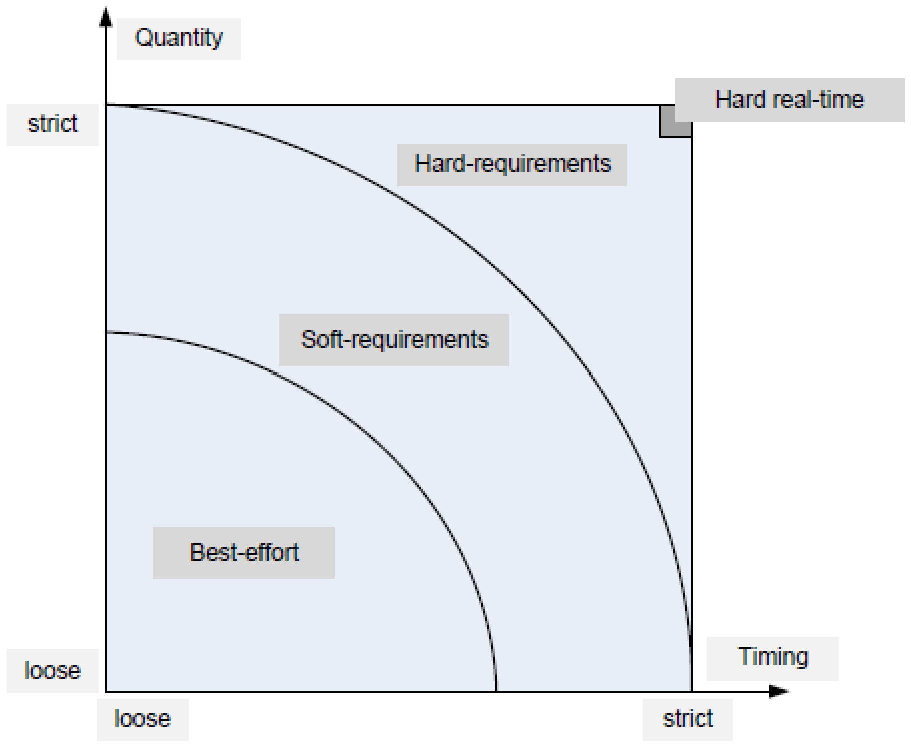
SCHEDULING

Cloud scheduling algorithms

- Scheduling → responsible for resource sharing at several levels:
  - A server can be shared among several virtual machines.
  - A virtual machine could support several applications.
  - An application may consist of multiple threads.
- A scheduling algorithm should be efficient, fair, and starvation-free.
- The objectives of a scheduler:
  - Batch system → maximize throughput and minimize turnaround time.
  - Real-time system → meet the deadlines and be predictable.
- Best-effort: batch applications and analytics.
- Common algorithms for best effort applications:
  - Round-robin, First-Come-First-Serve (FCFS).
  - Shortest-Job-First (SJF), Priority algorithms.

Cloud scheduling algorithms (cont'd)

- Multimedia applications (e.g., audio and video streaming)
  - Have soft real-time constraints.
  - Require statistically guaranteed maximum delay and throughput.
- Real-time applications have hard real-time constraints.
- Scheduling algorithms for real-time applications:
  - Earliest Deadline First (EDF).
  - Rate Monotonic Algorithms (RMA).
- Algorithms for integrated scheduling of several classes of applications:
  - Resource Allocation/Dispatching (RAD) .
  - Rate-Based Earliest Deadline (RBED).

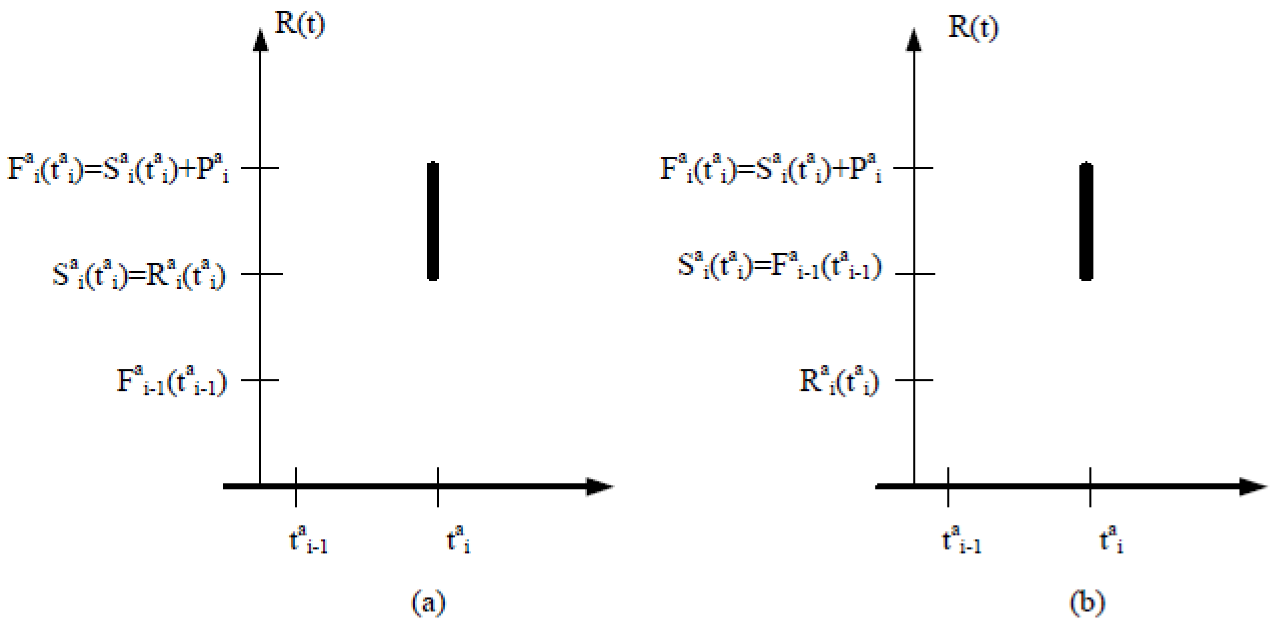


1. Best-effort policies → do not impose requirements regarding either the amount of resources allocated to an application, or the timing when an application is scheduled.
2. Soft-requirements policies → require statistically guaranteed amounts and timing constraints
3. Hard-requirements policies → demand strict timing and precise amounts of resources.

Fair Queuing, Start-time Fair Queuing, Borrowed Virtual Time

ALGORITHMS OF INTEREST

Fair queuing - schedule multiple flows through a switch



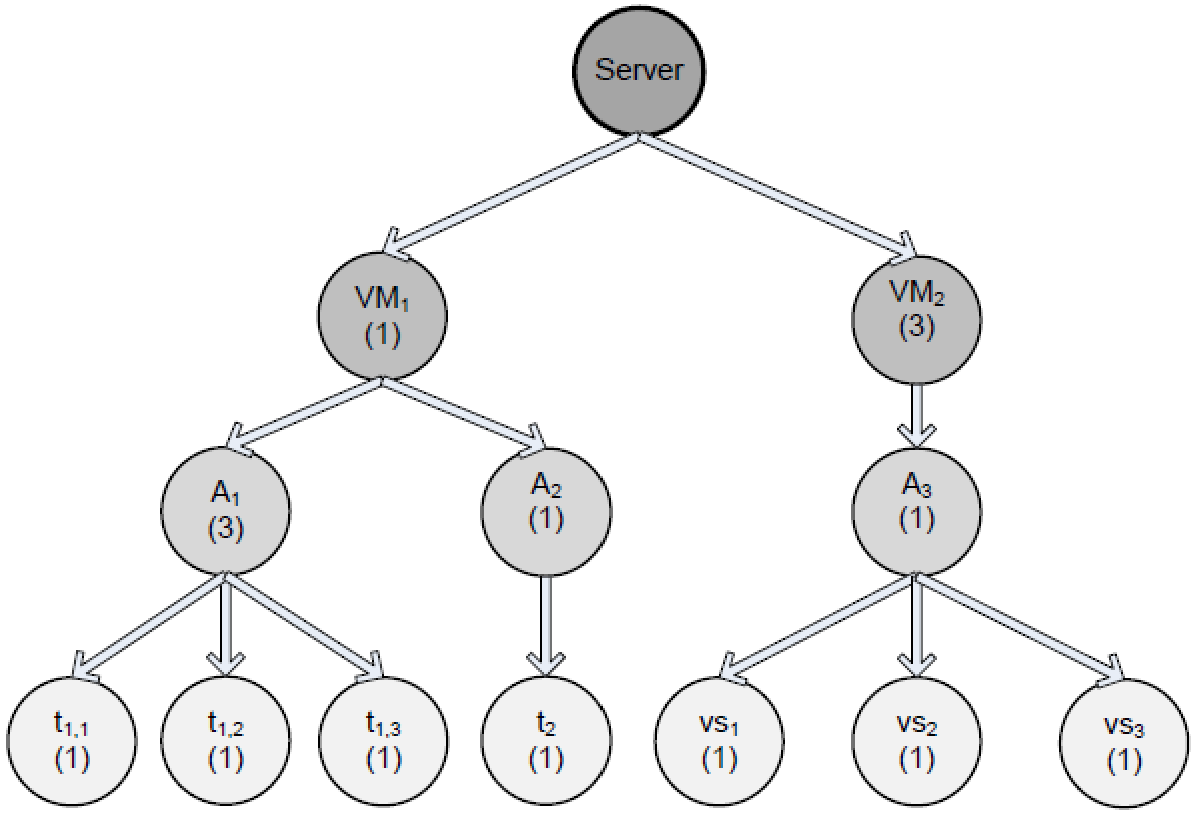
R(t)
$F^a_i(t^a_i)=S^a_i(t^a_i)+P^a_i$
$S^a_i(t^a_i)=R^a_i(t^a_i)$
$F^a_{i-1}(t^a_{i-1})$
t^a_{i-1}
t^a_i
(a)
R(t)
$F^a_i(t^a_i)=S^a_i(t^a_i)+P^a_i$
$S^a_i(t^a_i)=F^a_{i-1}(t^a_{i-1})$
$R^a_i(t^a_i)$
t^a_{i-1}
t^a_i
(b)

- The transmission of packet ***i*** of a flow can only start after the packet is available and the transmission of the previous packet has finished.
  - (a) The new packet arrives after the previous has finished.
  - (b) The new packet arrives before the previous one was finished.

Start-time fair queuing

- Organize the consumers of the CPU bandwidth in a tree structure.
- The root node is the processor and the leaves of this tree are the threads of each application.
  - When a virtual machine is not active, its bandwidth is reallocated to the other VMs active at the time.
  - When one of the applications of a virtual machine is not active, its allocation is transferred to the other applications running on the same VM.
  - If one of the threads of an application is not runnable then its allocation is transferred to the other threads of the applications.

The SFQ tree for Scheduling



- when two virtual machines VM1 and VM2 run on a powerful server

Borrowed Virtual time

- A latency-sensitive thread is allowed to *warp* back in virtual time.
- To make it appear earlier and thereby gain dispatch preference.
- It then effectively *borrows* virtual time from its future CPU allocation
  - thus does not disrupt long-term CPU sharing.
- Hence the name, *borrowed virtual time* scheduling.

Borrowed virtual time (BVT)

- Objective - support low-latency dispatching of real-time applications, and weighted sharing of CPU among several classes of applications.
- A thread ***i*** has
  - an effective virtual time, ***Ei***.
  - an actual virtual time, ***Ai***.
  - a virtual time warp, ***Wi***.
- The scheduler thread maintains its own scheduler virtual time (SVT) defined as the minimum actual virtual time of any thread.

BVT

- The threads are dispatched in the order of their effective virtual time, policy called the Earliest Virtual Time (EVT). Thread with earliest EVT is scheduled next

$$E \leftarrow A - (warpBack ? W : 0)$$

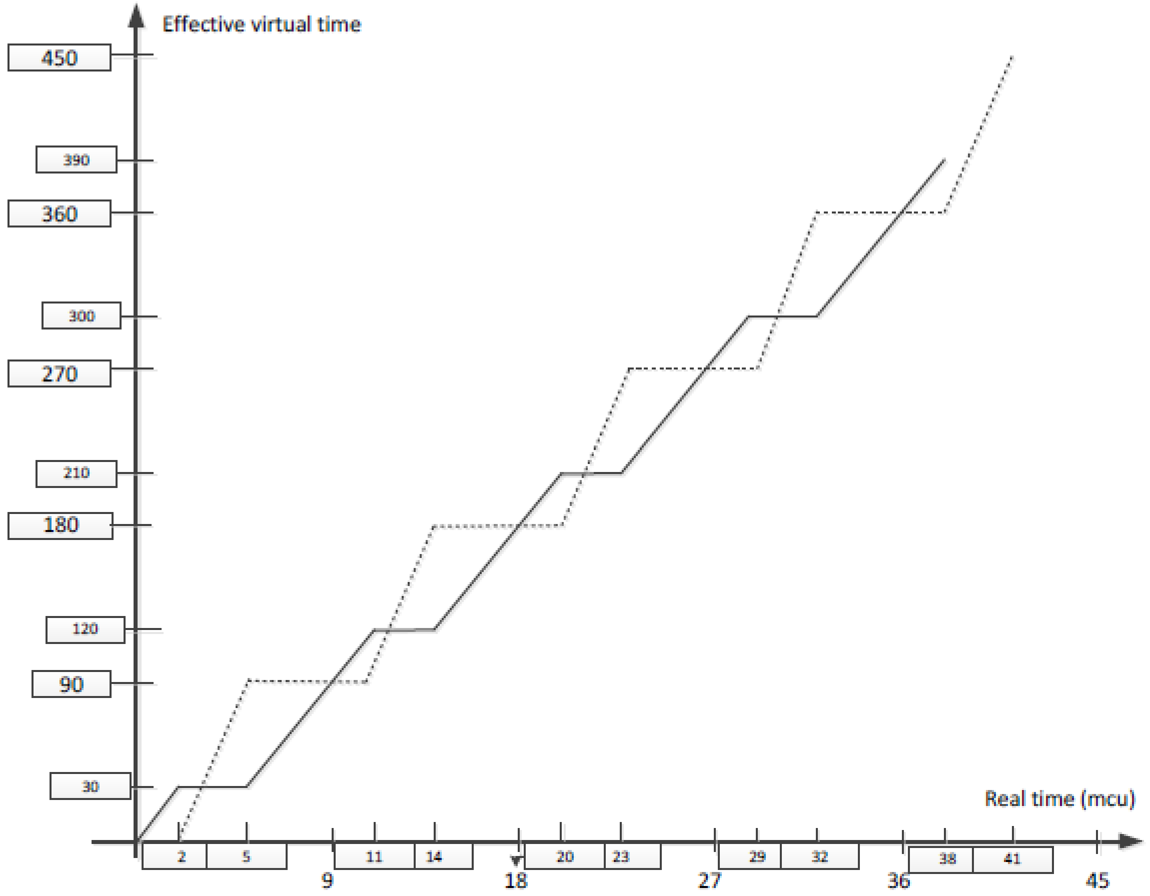
- Threads accumulate virtual time as they run. after thread runs for Δ time units, update

$$A_i \leftarrow A_i + (\Delta \times m_i)$$

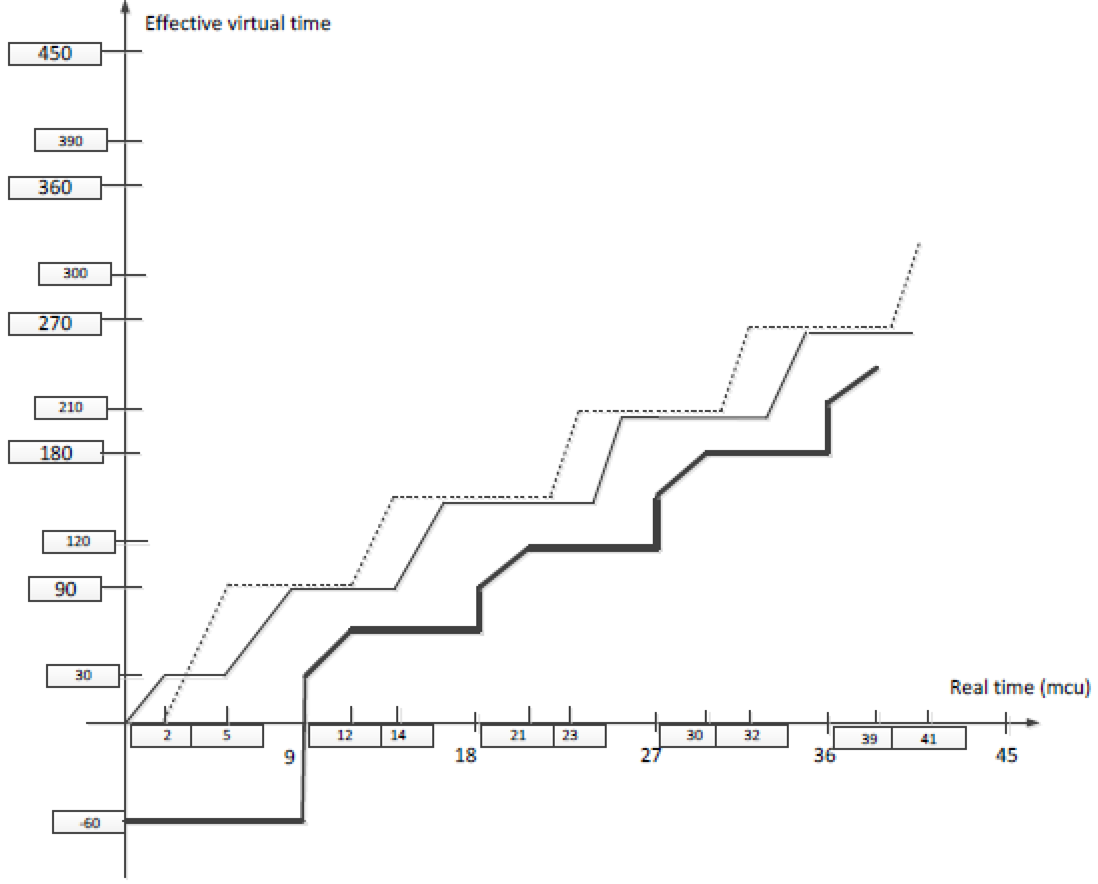
- m_i is the ‘inversely proportional weight’

BVT

- Can only switch every *C* time units to prevent thrashing
  - C: context switch allowance.
- Context switches are triggered by events such as:
  - the running thread is blocked waiting for an event to occur.
  - the time quantum expires
  - an interrupt occurs - running thread passes the waiting thread by C.
  - when a thread becomes runnable after sleeping.
- Threads can accumulate virtual time at different rates
  - Allow for weighted fair sharing of CPU
- To make sure that latency-sensitive threads are scheduled right away, give these threads high warp values
- Have limits on how much and how long can warp to prevent abuse



- The effective virtual time and the real time of the threads ***a*** (solid line) and ***b*** (dotted line) with weights *wa* = 2 *wb* when the actual virtual time is incremented in steps of 90 mcu.



- The effective virtual time and the real time of the threads ***a*** (solid line), ***b*** (dotted line), and the ***c*** with real-time constraints (thick solid line). Thread ***c*** wakes up periodically at times t=9, 18, 27, 36,..., is active for 3 units of time and has a time warp of 60 mcu.

Cloud scheduling subject to deadlines

- Hard deadlines → if the task is not completed by the deadline, other tasks which depend on it may be affected and there are penalties; a hard deadline is strict and expressed precisely as milliseconds, or possibly seconds.
- Soft deadlines → more of a guideline and, in general, there are no penalties; soft deadlines can be missed by fractions of the units used to express them, e.g., minutes if the deadline is expressed in hours, or hours if the deadlines is expressed in days.

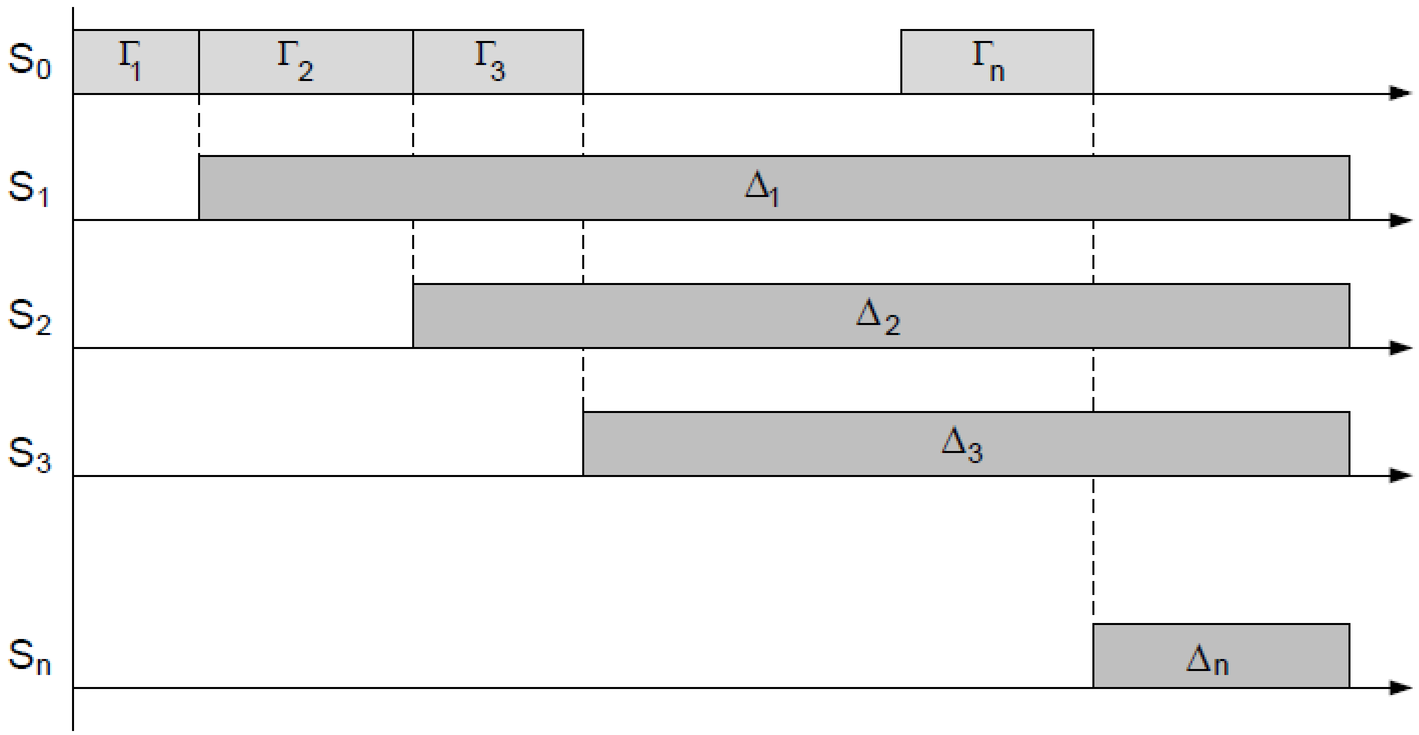
System Model

- Consider only aperiodic tasks with arbitrarily divisible workloads.
- Two important problems:
  - The order of execution of the tasks.
    - **Scheduling policies**
      - **FIFO**
      - **EDF**
      - **Maximum Workload derivative First**
  - ***The workload partitioning*** and the task mapping to worker nodes.

Workload partition rules

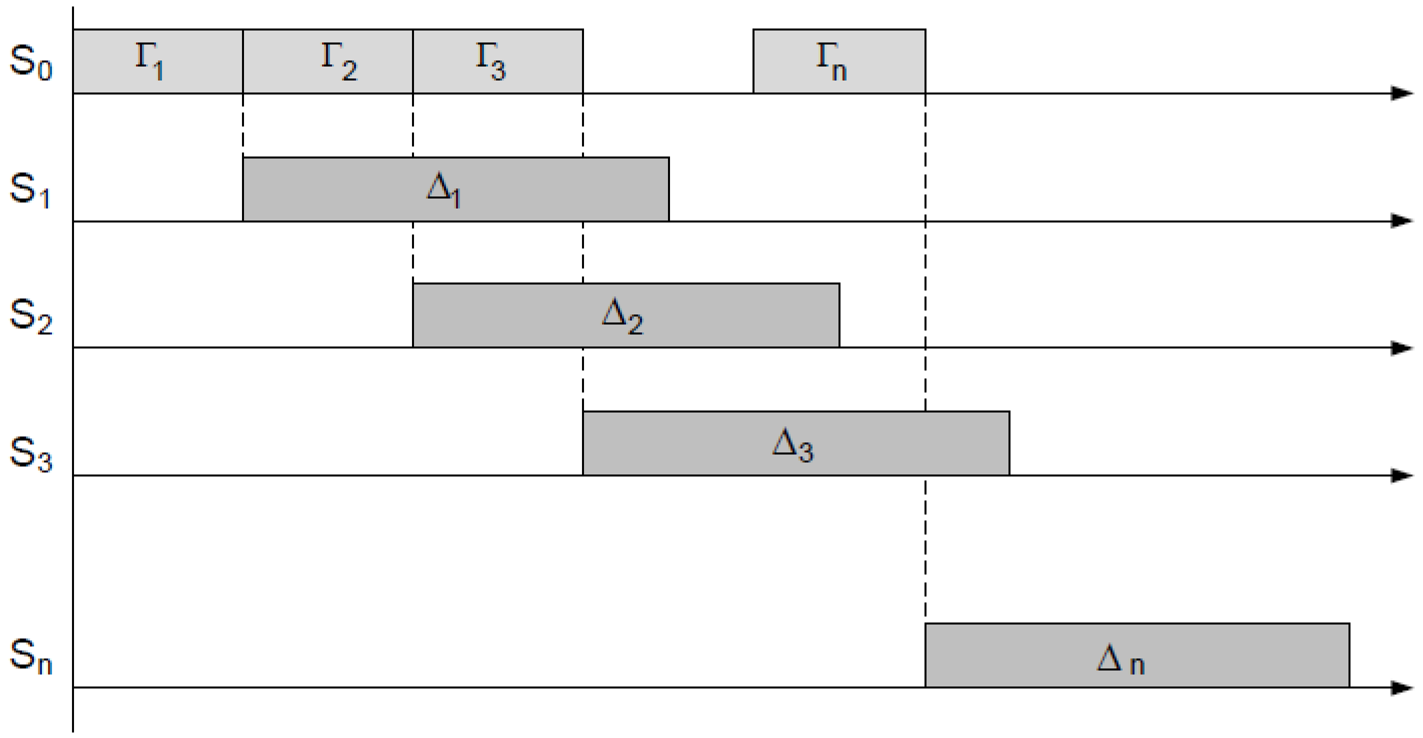
- Optimal Partitioning Rule (OPR) → the workload is partitioned to ensure
  - the earliest possible completion time and
  - all tasks are required to complete at the same time.
- Equal Partitioning Rule (EPR) → assigns an equal workload to individual worker nodes.

Optimum Partitioning Rule



- the algorithm requires worker nodes to complete execution at the same time. The head node, S0, distributes sequentially the data to individual worker nodes.

Equal Partitioning Rule



- The algorithm assigns an equal workload to individual worker nodes.

THANKS