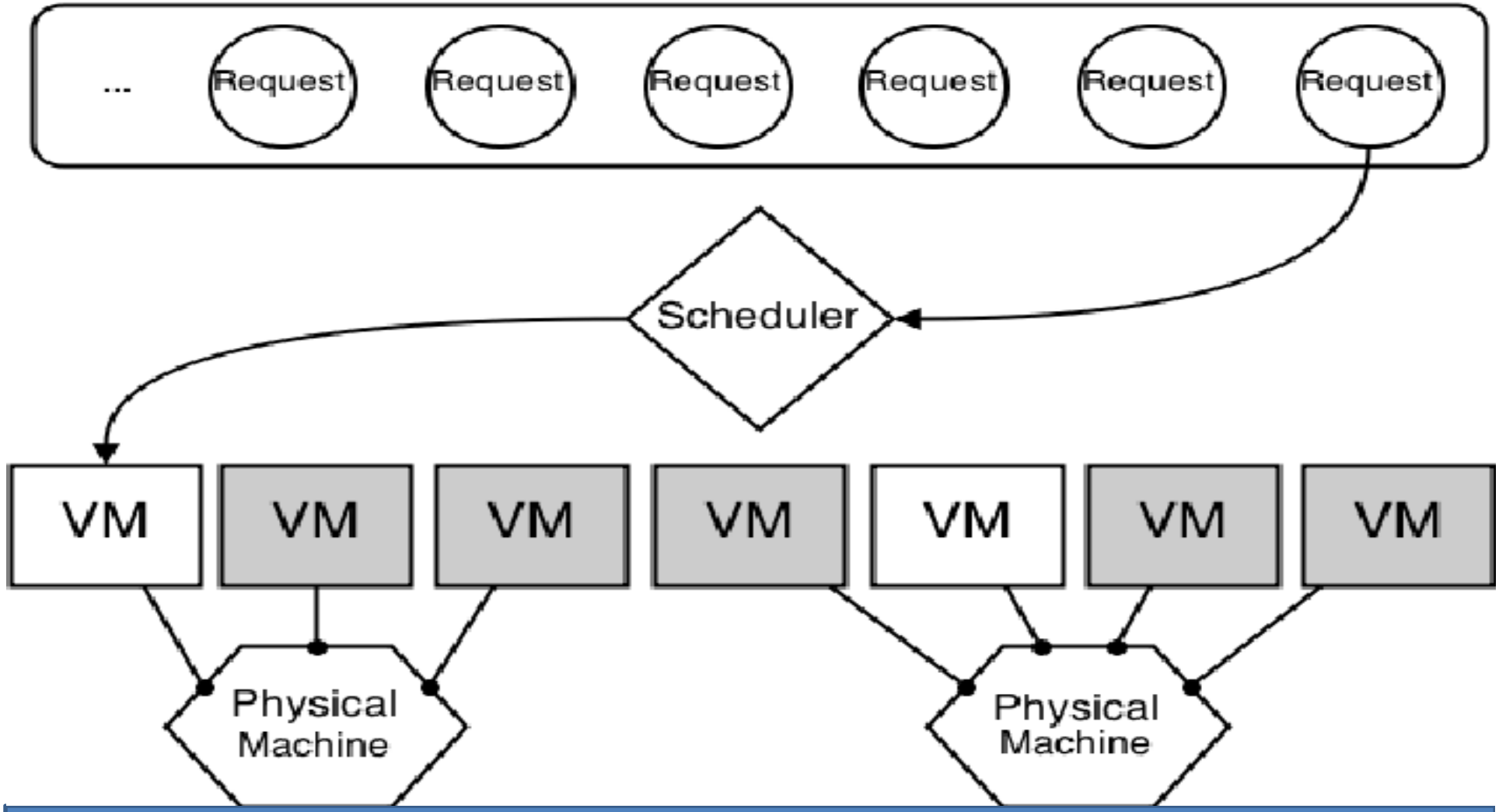
Cloud Resource Management and Scheduling



CHAPTER 05

Resource management

- Resource management is a critical function of any man-made system. It affects the three basic criteria for the evaluation of a system: performance, functionality, and cost.
- An inefficient resource management has a direct negative effect on performance and cost and an indirect effect on the functionality of a system.
- A cloud is a complex system with a very large number of shared resources subject to unpredictable requests.

Resource management

- The centralized control is unlikely to provide continuous service and performance guarantees.
- Autonomic policies are of great interest due to the scale of the system, the large number of service requests, the large user population, and the unpredictability of the load.
- 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.

Motivation

- 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. E.g., for Web services subject to seasonal spikes.

Motivation

- The strategies for resource management for IaaS, PaaS, and SaaS are different.

Policies and mechanisms for resource management

- A policy typically refers to the principal guiding decisions, whereas mechanisms represent the means to implement policies.
- Cloud resource management policies can be loosely grouped into five classes:
  1. Admission control.
  2. Capacity allocation.
  3. Load balancing.
  4. Energy optimization.
  5. Quality-of-service (QoS) guarantees

Policies and mechanisms for resource management

- A policy typically refers to the principal guiding decisions, whereas mechanisms represent the means to implement policies.
- Cloud resource management policies can be loosely grouped into five classes:
  1. Admission control.
  2. Capacity allocation.
  3. Load balancing.
  4. Energy optimization.
  5. Quality-of-service (QoS) guarantees

Cloud resource management (CRM) policies

1. Admission control → prevent the system from accepting workload in violation of high-level system policies.
2. Capacity allocation → allocate resources for individual activations of a service.
3. Load balancing → distribute the workload evenly among the servers.
4. Energy optimization → minimization of energy consumption.
5. 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

- Allocation techniques in computer clouds must be based on a disciplined approach rather than ad hoc methods. The four basic mechanisms for the implementation of resource management policies are:
  - *Control theory*
  - *Machine learning*
  - *Utility-based mechanisms*
  - *Market-oriented/economic mechanisms*

Mechanisms for the implementation of resource management policies

- Control theory → deals with the behavior of dynamical systems with inputs, and how their behavior is modified by feedback.
- Machine learning → is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
- Utility-based → model allows clients to choose services, tools, environments, and resources from an on-demand platform as plug-and-play utilities.
- Market-oriented/economic → "A business approach or philosophy that focuses on identifying and meeting the stated or hidden needs or wants of customers.

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.

Table 6.1 The normalized performance and energy consumption function of the processor speed. The performance decreases at a lower rate than does the energy when the clock rate decreases.

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.

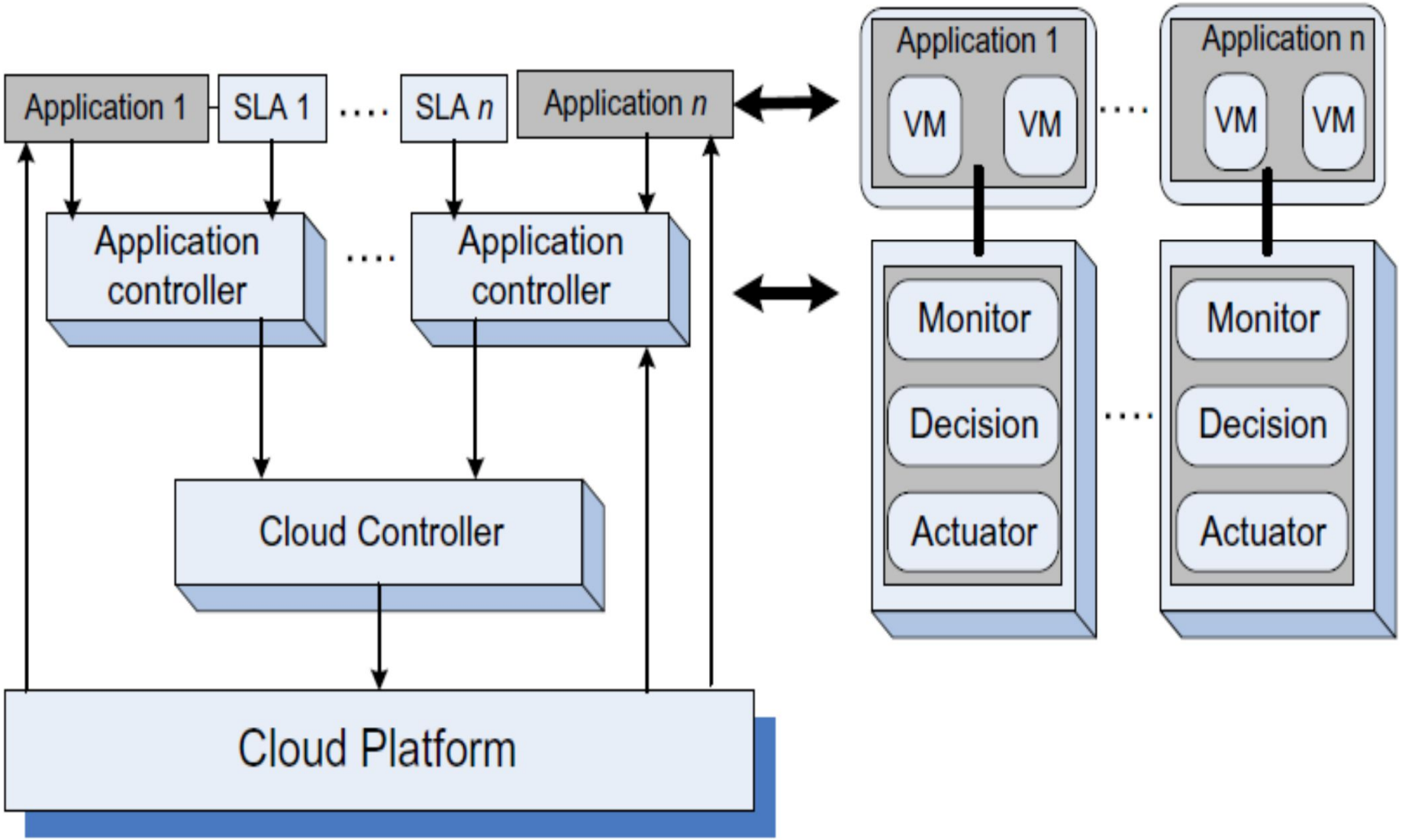
Control theory application to cloud resource management (CRM)

- 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.

Two-level cloud controller

- It is a two-level resource allocation architecture based on control theory concepts for the entire cloud.
- The automatic resource management is based on two levels of controllers, one for the service provider and one for the application.
- The actions consist of allocation/deallocation of one or more virtual machines.
- Application controllers → determine need of additional resources.
- Cloud controllers → decide requests for resources and allocates the physical resources.

Two-level cloud controller



Two-level cloud controller

- The elements involved in a control system are sensors, monitors, and actuators. The sensors measure the parameter(s) of interest, then transmit the measured values to a monitor, which determines whether the system behavior must be changed, and, if so, it requests that the actuators carry out the necessary actions.
- The parameter used for admission control policy is the current system load; when a threshold, e.g., 80%, is reached, the cloud stops accepting additional load.

Feedback control based on dynamic thresholds

- A *threshold* is the value of a parameter related to the state of a system that triggers a change in the system behavior.
- Thresholds are used in control theory to keep critical parameters of a system in a predefined range.
- The threshold could be *static*, defined once and for all, or it could be *dynamic*.

Feedback control based on dynamic thresholds

- A dynamic threshold could be based on an average of measurements carried out over a time interval, a so-called *integral control*.
- The dynamic threshold could also be a function of the values of multiple parameters at a given time.
- To maintain the system parameters in a given range, a *high* and a *low* threshold are often defined.
- The two thresholds determine different actions; for example, a high threshold could force the system to limit its activities and a low threshold could encourage additional activities.

Proportional thresholding

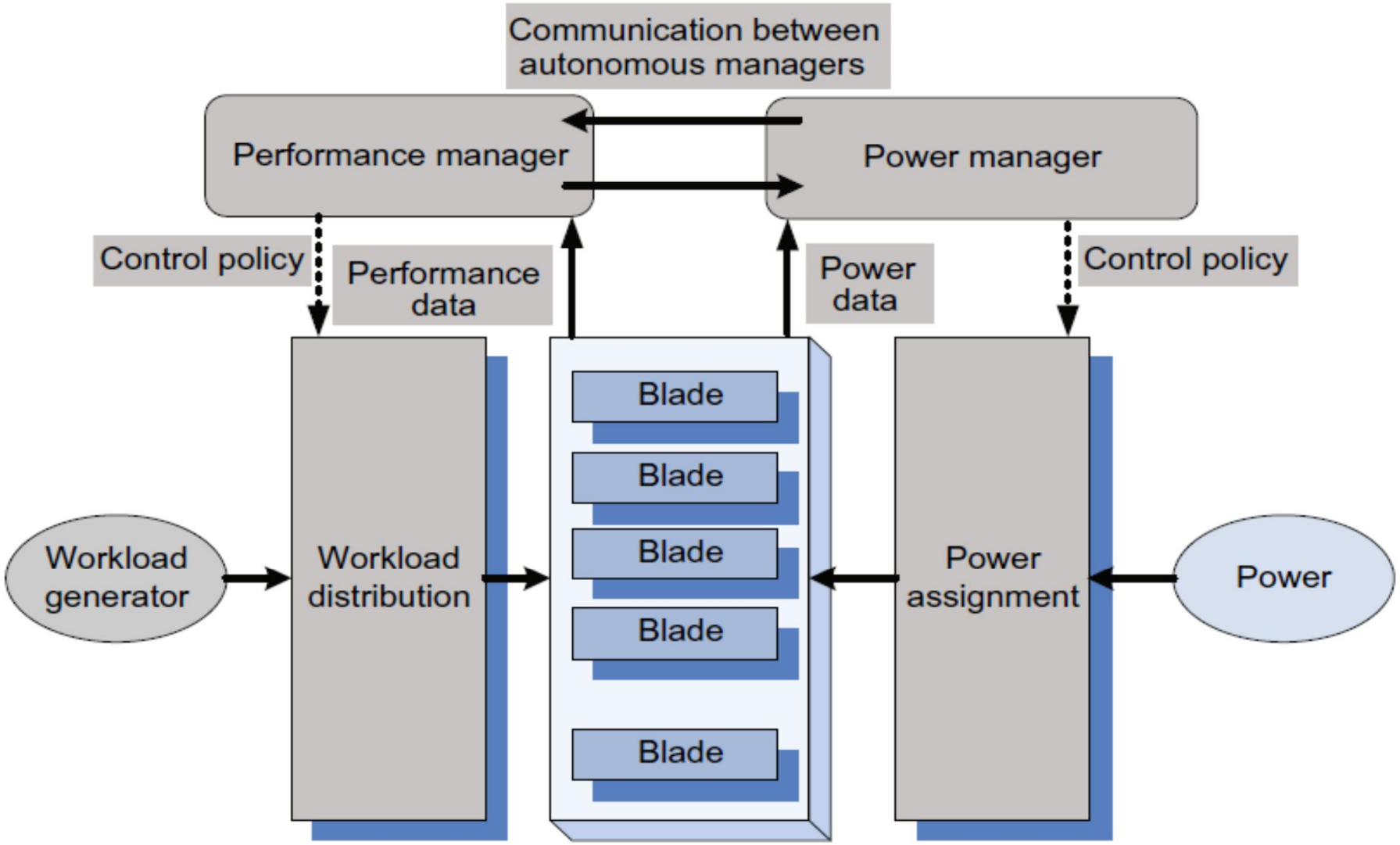
- *Thresholding used for IaaS* delivery model, and a strategy for resource management is called *proportional thresholding.*

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.

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 according to the power management policy.
- Use standard software systems.



Autonomous performance and power managers cooperate to ensure SLA prescribed performance and energy optimization. They are fed with performance and power data and implement the performance and power management policies, respectively.

A utility-based model for cloud-based Web services

- **Utility-based model** allows clients to choose **services**, tools, environments, and resources from an on-demand platform as plug-and-play utilities.
- It is a **service** provisioning **model** in which a **service provider** makes **computing** resources and infrastructure management available to the **customer** as needed, and **charges** them for specific usage rather than a flat rate.

A utility-based model for cloud-based Web services

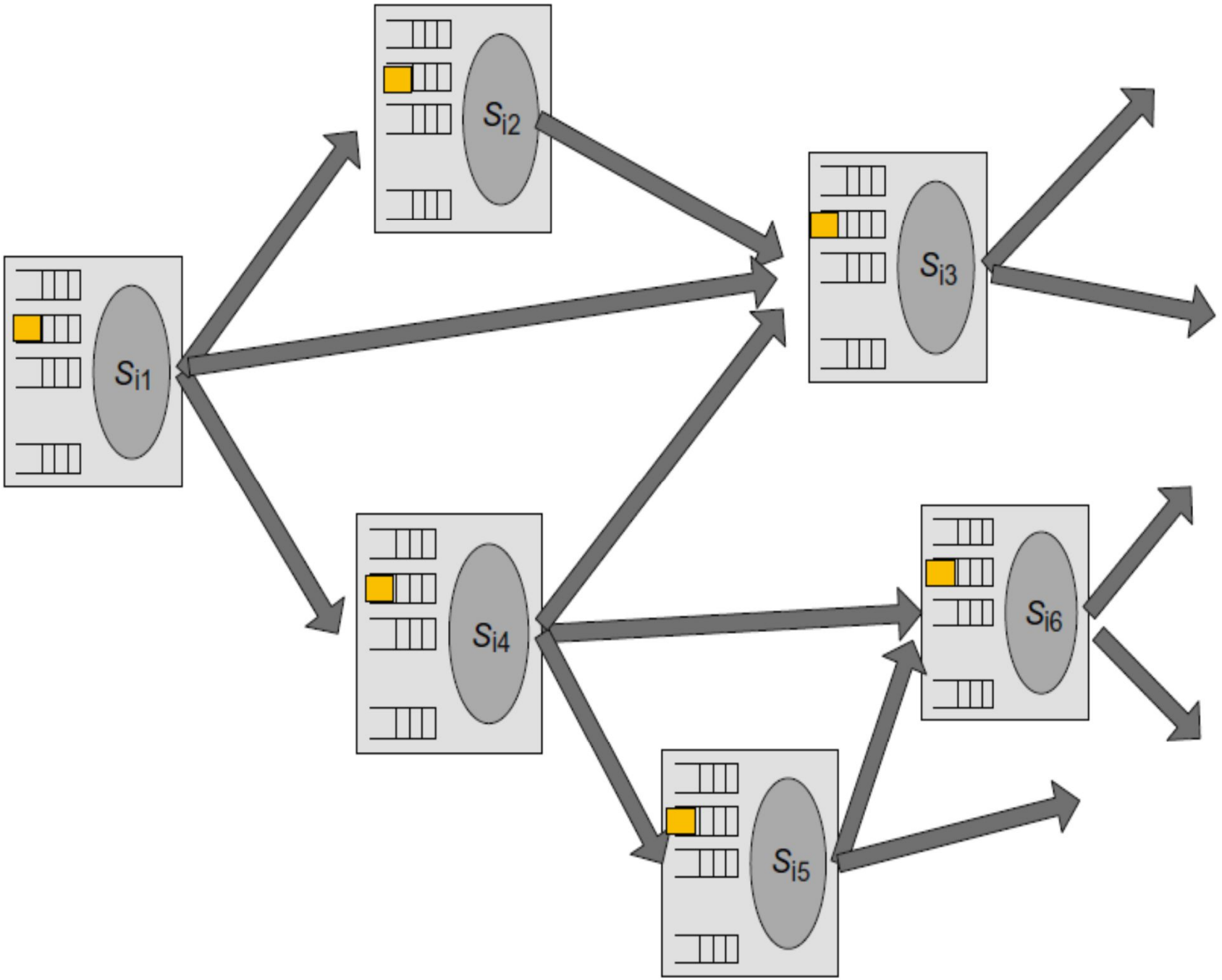
- A **utility function** is a representation to define individual preferences for goods or services beyond the explicit monetary value of those goods or services. it is a calculation for how much someone desires something, and it is relative.
- A utility function relates the “benefits” of an activity or service with the “cost” to provide the service.
- For example, the benefit could be revenue and the cost could be the power consumption.

A utility-based model for cloud-based Web services

- The goal is to maximize the total profit computed as the difference between the revenue guaranteed by an SLA and the total cost to provide the 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.

A utility-based model for cloud-based Web services

- The system is modeled as a network of queues with multi-queues for each server.
- Cloud provides *K* different classes of service, each class *k* involving N_K applications.


S_{i1}
S_{i2}
S_{i3}
S_{i4}
S_{i5}
S_{i6}

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.

Resource bundling

- In the context of cloud computing, an auction is. the allocation of resources to the highest bidder

Combinatorial Auctions

- Auctions in which participants can bid on combinations of items, or packages, are called combinatorial auctions.
- Such auctions provide a relatively simple, scalable, and tractable solution to cloud resource allocation.
- 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.

Combinatorial Auctions

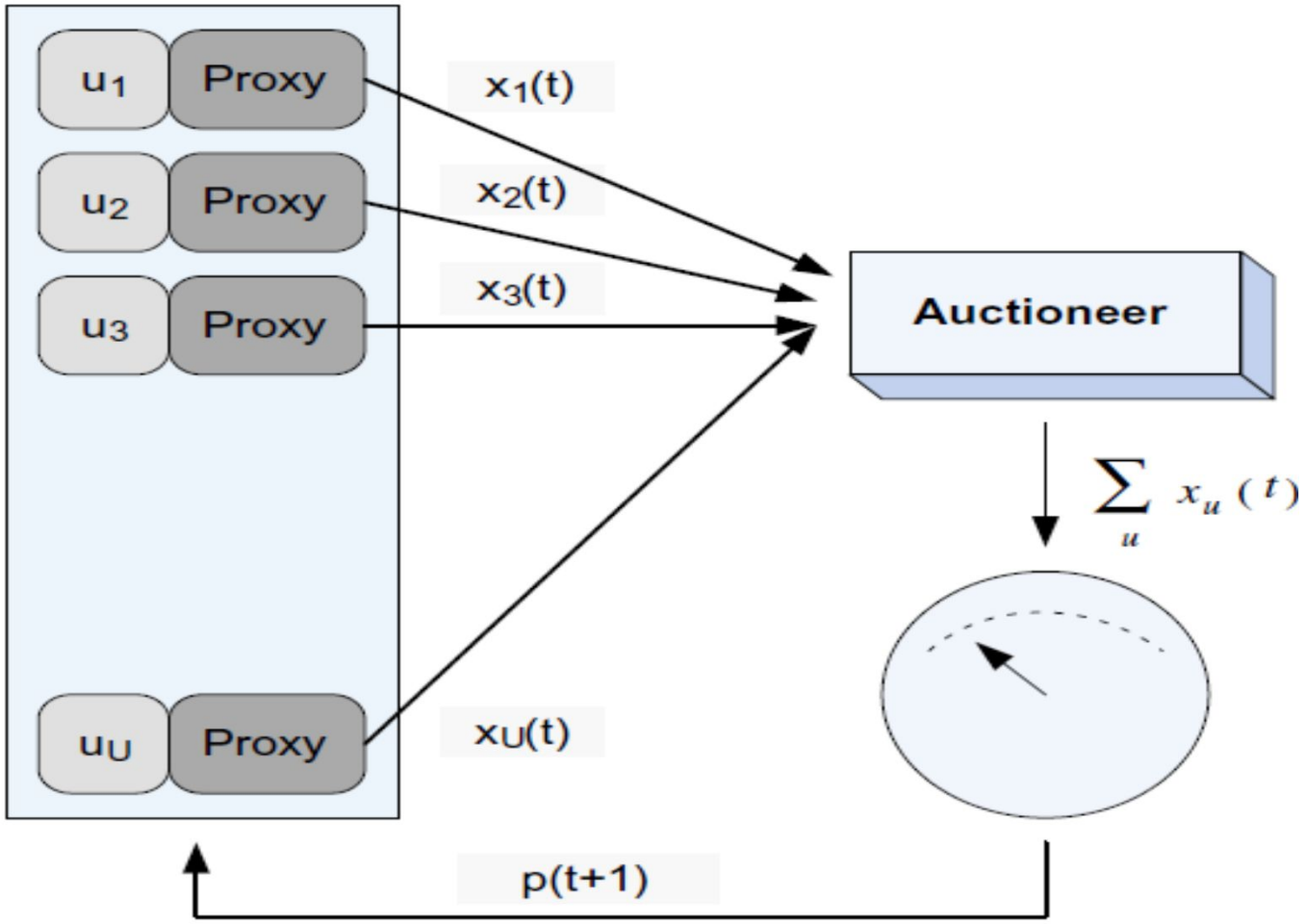
- Given a set of bids in a **combinatorial auction**, find an allocation of items to bidders—including the possibility that the auctioneer retains some items—that maximizes the auctioneer's revenue.

AScending Clock Auction (ASCA)

- This describes auctions where bidders place bids of progressively higher amounts, aiming to **outbid** each other. The bidder who places the highest bid by the end of the auction wins.
- In the ASCA algorithm [333] the participants at the auction specify the resource and the quantities of that resource offered or desired at the price listed for that time slot.

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.



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

Scheduling algorithms for clouds

- Scheduling → responsible for resource sharing/multiplexing 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.

Scheduling algorithms for clouds

- The objectives of a scheduler:
  - Batch system → maximize throughput and minimize turnaround time.
  - Real-time system → meet the deadlines and be predictable.
- Some schedulers are *preemptive*, allowing a high-priority task to interrupt the execution of a lower-priority one; others are *non-preemptive*.

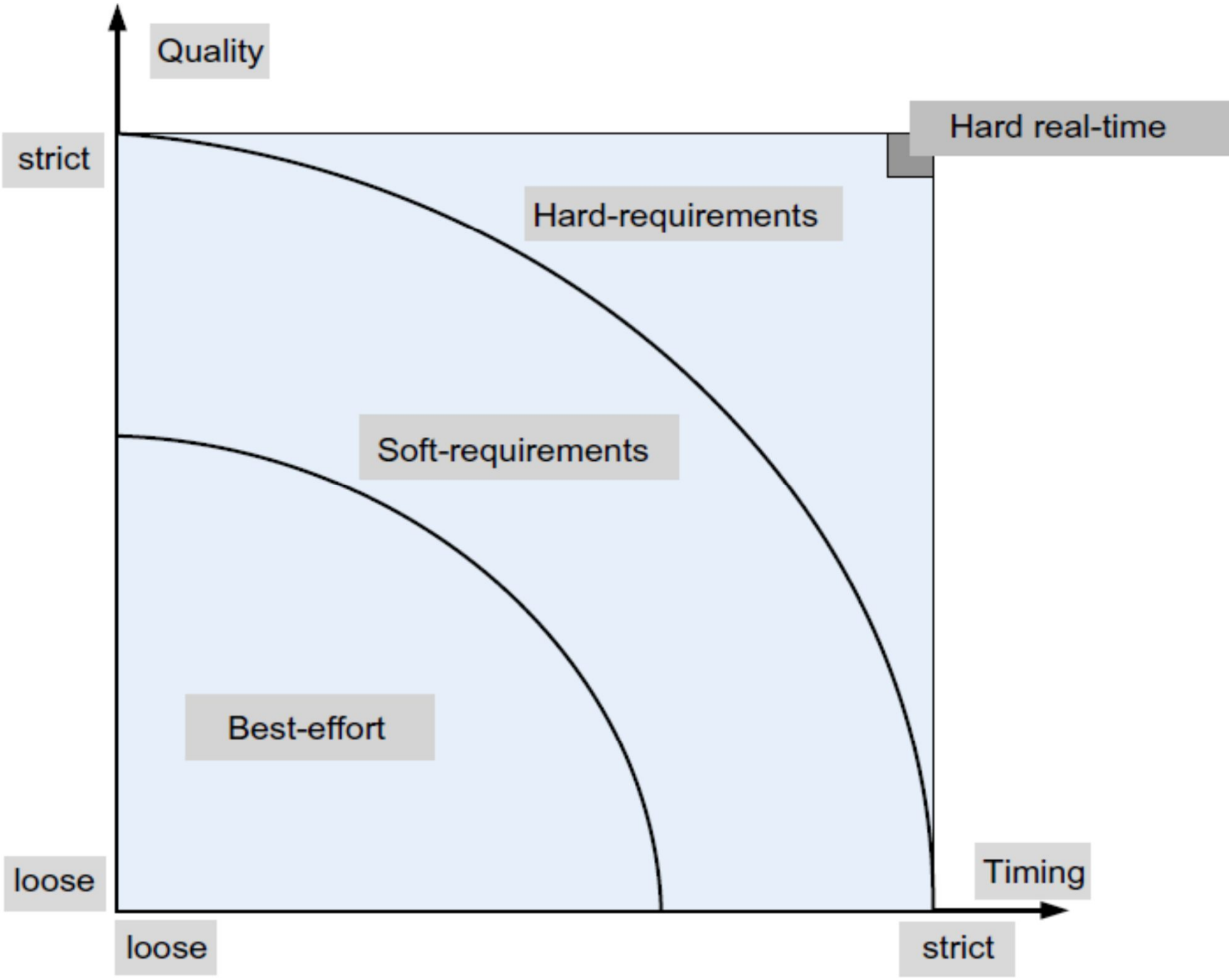
Scheduling algorithms for clouds

- Two distinct dimensions of resource management must be addressed by a scheduling policy:

(a) the amount or quantity of resources allocated
(b) the timing when access to resources is granted

Broad classes of resource allocation requirements

- Best effort
- Soft requirements
- Hard requirements.


Quality
strict
Hard real-time
Hard-requirements
Soft-requirements
Best-effort
loose
Timing
loose
strict

Broad classes of resource allocation requirements

- 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.
- Soft-requirements allocation policies require statistically guaranteed amounts and timing constraints;
- Hard-requirements allocation policies demand strict timing and precise amounts of resources. Hard-real time systems are the most challenging because they require strict timing and precise amounts of resources.

Broad classes of resource allocation requirements

- The quality-of-service (QoS) requirements differ for different classes of cloud applications and demand different scheduling policies.
- Best-effort applications such as batch applications and analytics do not require QoS guarantees.
- Multimedia applications such as audio and video streaming have soft real-time constraints and require statistically guaranteed maximum delay and throughput.
- Real-time applications have hard real-time constraints.

Common algorithms for best effort applications:

- Round-robin. [fair and starvation-free]
- First-Come-First-Serve (FCFS).
- Shortest-Job-First (SJF).
- Priority algorithms.

Other scheduling algorithms

- Scheduling algorithms for real-time applications:
  - Earliest Deadline First (EDF).
  - Rate Monotonic Algorithms (RMA).
- Algorithms for Integration of scheduling for the three classes of application
  - Resource Allocation/Dispatching (RAD) .
  - Rate-Based Earliest Deadline (RBED).

Fair queuing

- Computing and communication on a cloud are intimately related.
- Interconnection networks allow cloud servers to communicate with one another and with users.
- These networks consist of communication links of limited bandwidth & switches/routers/gateways of limited capacity.
- When the load exceeds its capacity, a switch starts dropping packets because it has limited input buffers for the switching fabric and for the outgoing links, as well as limited CPU cycles.

Fair queuing

- A switch must handle multiple flows and pairs of source-destination endpoints of the traffic. Thus, a scheduling algorithm has to manage several quantities at the same time: the *bandwidth*, the amount of data each flow is allowed to transport; the *timing* when the packets of individual flows are transmitted; and the *buffer space* allocated to each flow.

Fair queuing

- A first strategy to avoid network congestion is to use a FCFS scheduling algorithm. The advantage of the FCFS algorithm is a simple management of the three quantities: bandwidth, timing, and buffer space.
- Nevertheless, the FCFS algorithm does not guarantee fairness; greedy flow sources can transmit at a higher rate and benefit from a larger share of the bandwidth

Fair queuing

- To address this problem, a fair queuing algorithm proposed, in which separate queues are maintained by a switch for each flow, and the queues are serviced in a round-robin manner.
- "Fair Queuing" is an attempt to give the flows equal shares, at least within the limits of actual demand.
- The fairness of bandwidth allocation is achieved by adopting *bit-by-bit round-robin (BR)* strategy, in which, a single bit from each queue is transmitted and the queues are visited in a round-robin fashion.

Resource management and dynamic application scaling

- The demand for computing resources, such as CPU cycles, primary and secondary storage, and network bandwidth, depends heavily on the volume of data processed by an application.
- The demand for resources can be a function of the time of day, can monotonically increase or decrease in time, or can experience predictable or unpredictable peaks.

Resource management and dynamic application scaling

- For example, a new Web service will experience a low request rate when the service is first introduced and the load will exponentially increase if the service is successful.
- A service for income tax processing will experience a peak around the tax filling deadline. [Aadhar linking service with deadline]
- The elasticity of a public cloud can supply required amount of resources to an application by adopting dynamic scaling methods.

How scaling can be implemented in cloud

- *Scaling modes*
  - *Vertical scaling*
  - *Horizontal scaling*

Vertical scaling

- *Vertical scaling* keeps the number of VMs of an application constant, but increases the amount of resources allocated to each one of them.
- This can be done either by migrating the VMs to more powerful servers or by keeping the VMs on the same servers but increasing their share of the CPU time.

Horizontal scaling

- *Horizontal scaling* is the most common mode of scaling on a cloud; it is done by increasing the number of VMs as the load increases and reducing the number of VMs when the load decreases.
- Often, this leads to an increase in communication bandwidth consumed by the application.
- Load balancing among the running VMs is critical to this mode of operation.

Horizontal scaling

- For a very large application, multiple load balancers may need to cooperate with one another. In some instances the load balancing is done by a front-end server that distributes incoming requests of a transaction-oriented system to back-end servers.
- To scale up and down a compute-intensive application, a good strategy is to increase or decrease the number of VMs or instances.

Automatic VM scaling

- *Automatic VM scaling* uses predefined metrics, e.g., CPU utilization, to make scaling decisions.
- Automatic scaling requires *sensors* to monitor the state of VMs and servers; *controllers* make decisions based on the information about the state of the cloud.

Examples for Scaling

- *AWS Cloud Watch*
- *Elastic Load Balancing* service
- *Elastic Beanstalk*


DON'T LIMIT YOUR CHALLENGES -
CHALLENGE YOUR LIMITS.
CHARGING
LIFE