Unit-4

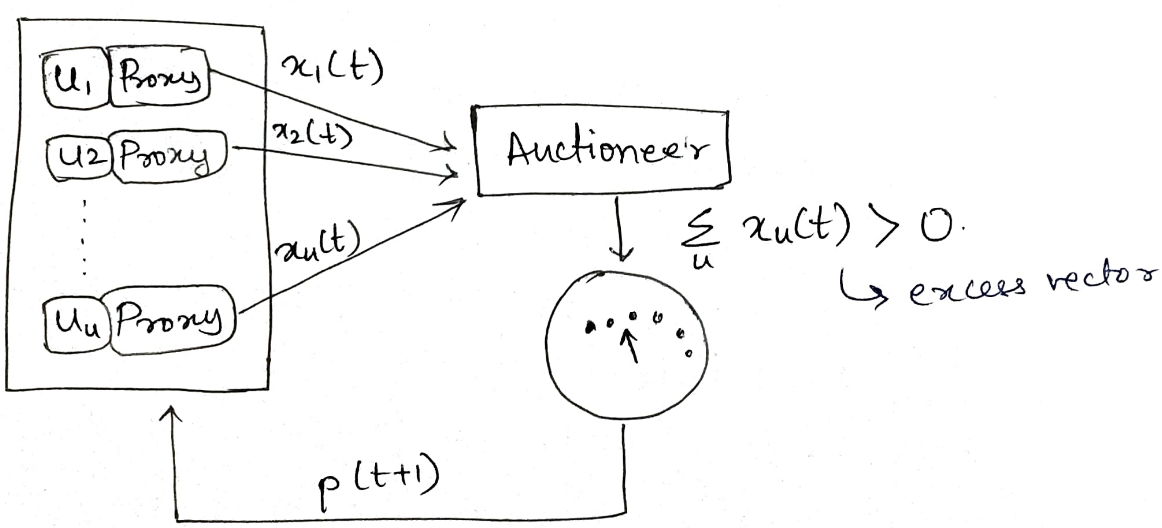
1. Describe using Schematics, the ASCA Combination Auction algo.

Ans:

* Ascending Clock Auction (ASCA) → the current price for each resource is presented 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.

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



* If all its components are -ve, the auction stops.
* -ve components mean demand doesn't exceed the offer.

~~(2) Illustrate how coordination b/w autonomic system can be utilized for cloud resource management.~~

(3) Policies for Cloud Resource Management.

→ Policies can be grouped into 5 classes.

1. Admission control
2. Capacity allocation
3. Load balancing
4. Energy optimization
5. Quality-of-Service (QoS) guarantees.

→ 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.

→ QoS Guarantees - ability to satisfy timing or other conditions specified by a Service Level Agreement (SLA).

* Mechanisms to implement CRM:

1. Control theory - uses the feedback to guarantee system stability and predict transient behaviour.
2. Machine Learning - doesn't need a performance model of system.
3. Utility Based - require a performance model and a mechanism to correlate user-level performance with cost.
4. Market-oriented/economic - don't require a model of the system, eg., combinatorial auctions for bundles of resources.

4) Conditions of pricing and allocation algorithms

→ A pricing & allocation algorithm partitions the set of users in two disjoint sets, winners & losers.

→ Conditions are as follows:

1. Be computationally tractable.
↳ Traditional combinatorial auction (VLG) are not computationally tractable

2. Scale well - given the scale of the system & the no. of requests for service, scalability is a necessary condⁿ.

3. Be objective - partitioning in winners & 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 a loser.

4. Be fair - make sure that the prices are uniform, all winners within a given resource pool pay the same price

5. Indicate clearly at the end of the auction the unit prices for each resource pool.

6. Indicate clearly to all participants the relationship b/w the supply & the demand in the system.

5) Start-Time Fair Queuing (Describe + Numerical

→ Organize the consumers of the CPU bandwidth in a tree structure

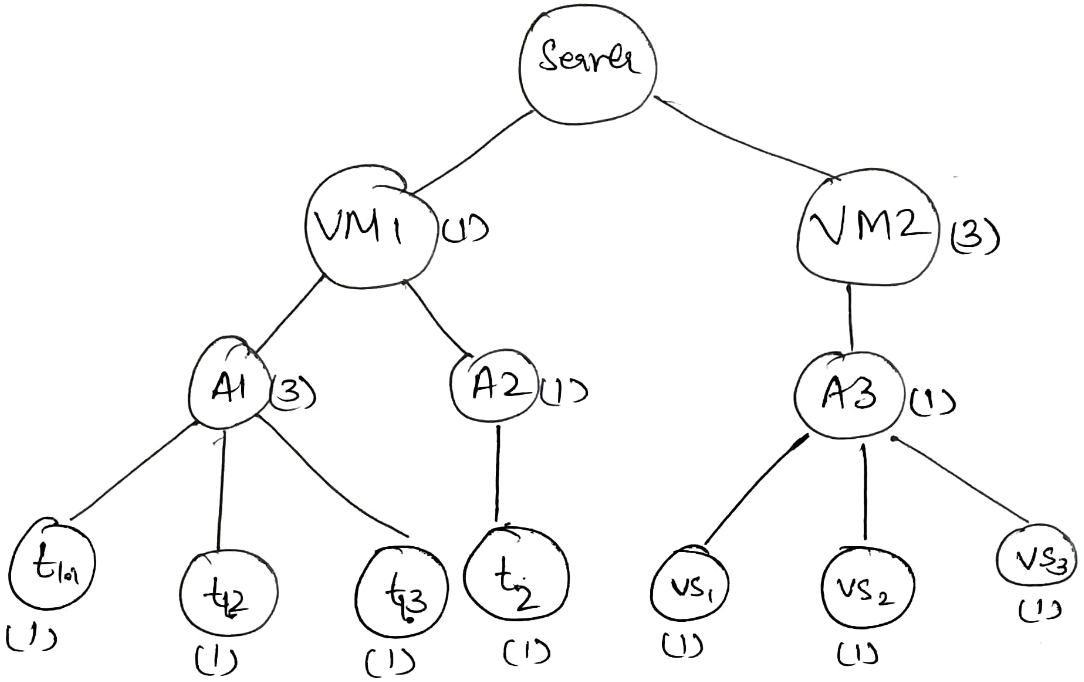
→ The root node is the processor & the leaves of this tree are the threads of each application.

* When a virtual machine is not active, its bandwidth is reallocated to the other VMs active at the time

* When one ~~app~~ of the appl's of a VM is not active, its allocation is transferred to the appl's running on the same VM.

* If one of the threads of an applⁿ is not runnable, then its allocation is transferred to the other threads of the applⁿ

* Eg: SFQ tree for scheduling when two virtual machines VM_1 & VM_2 run on a powerful server.



~~Two Level Control architecture for~~

→ SFQ scheduler follows some rules –

1. The threads are serviced in the order of their virtual startup time; ties are broken arbitrarily.
2. The virtual startup time of the i-th activation of thread x is:

$$\boxed{S_x^i(t) = \max\left[V(\tau^j), F_x^{(i-1)}(t)\right] \text{ and } S_x^0 = 0}$$

The cond^n for thread i to be started is that thread (i-1) has finished & that scheduler is active.

3. The virtual finish time of the i-th activation of thread x is:

$$F_x^i(t) = S_x^i(t) + \frac{q}{w_x}$$

w_x → weight of thread
q → quantum of scheduler.

4. The virtual time of all threads is initially 0, $V_x^0 = 0$

$$V(t) = \begin{cases} \text{virtual start time of thread in service, if CPU is busy.} \\ \text{max. finish virtual time of an time, if CPU is idle} \end{cases}$$

Timing Diagrams (OPR & EPR)

→ **OPR** – Optimal Partitioning Rule

* refers to the execution time.
* workload is partitioned to ensure earliest possible completion time & all tasks are required to complete at the same time

$$n^{min} = \left\lceil \frac{\ln \gamma}{\ln \beta} \right\rceil$$

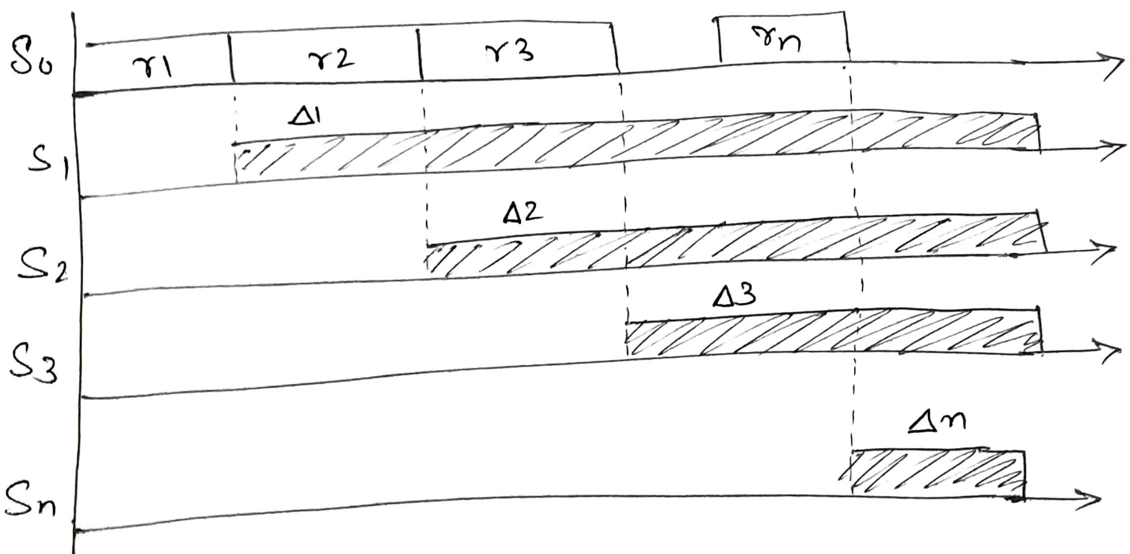
* The head node S_0 distributes sequentially the data to individual worker nodes.

→ **EPR** – Equal Partitioning Rule

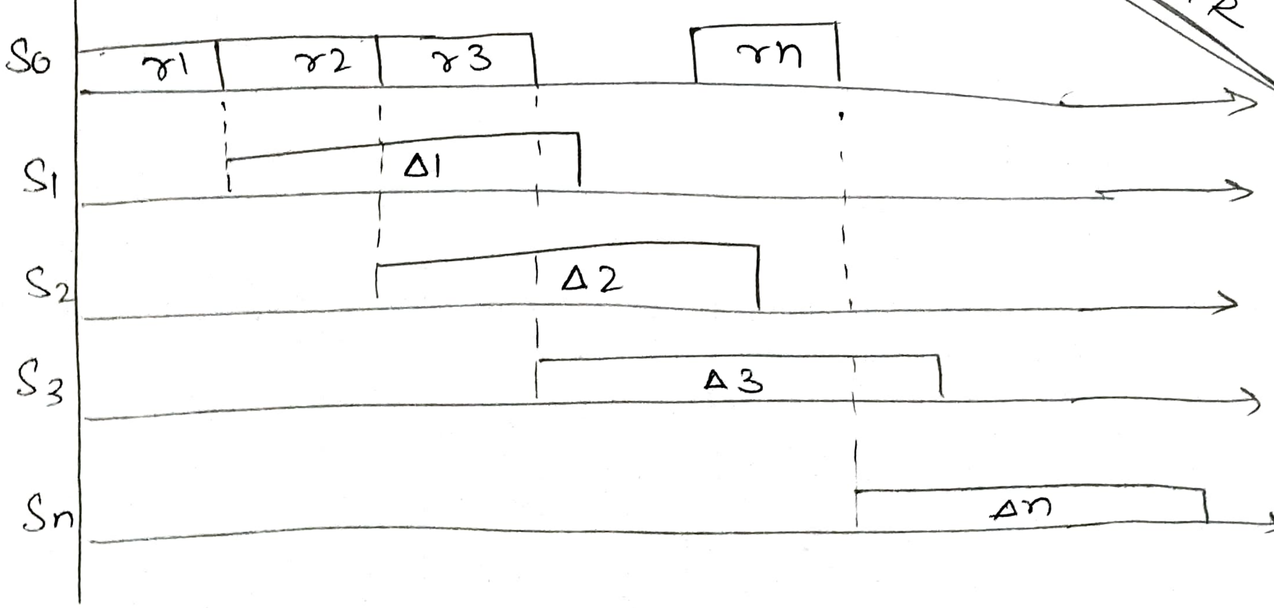
* assigns an equal workload to individual worker nodes.

$$n^{min} = \left\lceil \frac{\sigma \times p}{A + D - t_0 - \sigma + \tau} \right\rceil$$

OPR



EPR



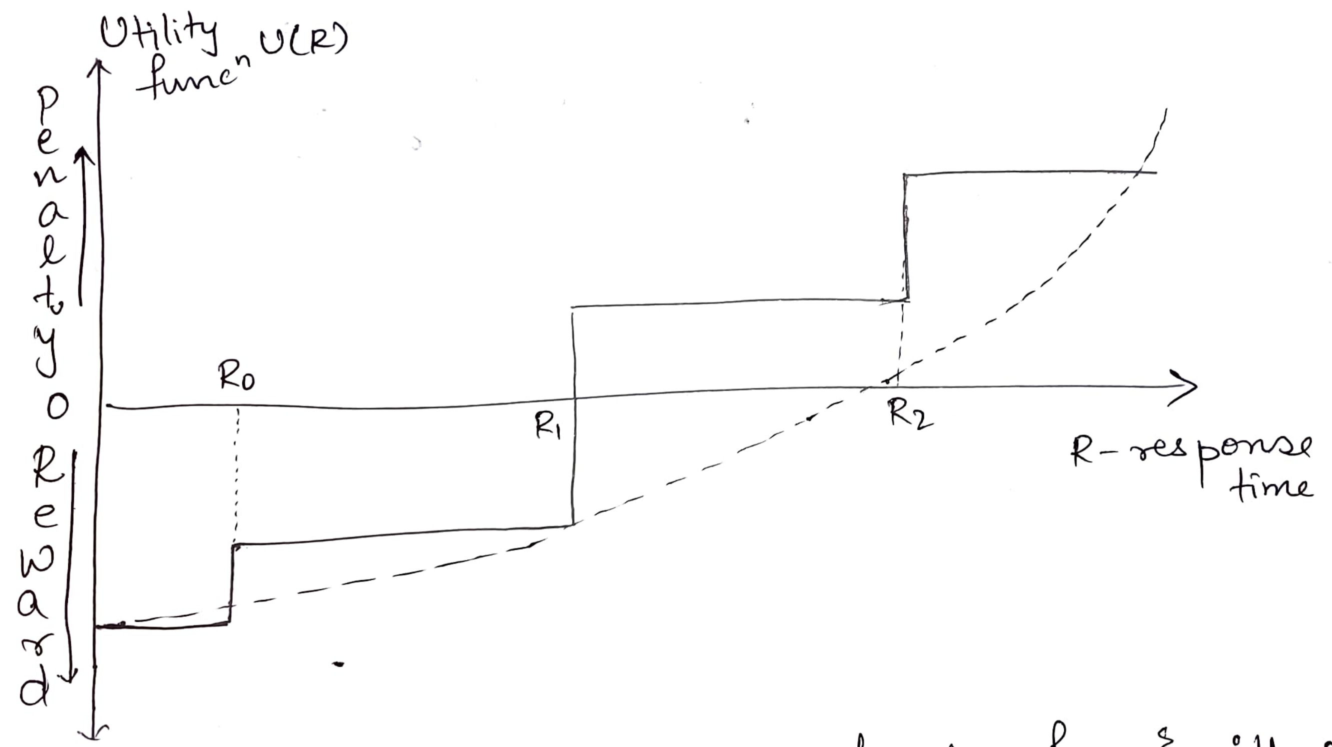
(8) Illustrate utility func^n when the performance metric is response time.

→ Utility function $U_{pp}(R,P)$ is a func^n of the response time R and the power P and it can be of the form:

$$U_{pp}(R,P) = U(R) - \epsilon \times P.$$

or

$$U_{pp}(R,P) = \frac{U(R)}{P}.$$



* The utility func^n is a series of step func^{n}s with jumps corresponding to the response time, $R = R_0 | R_1 | R_2$, when the ~~...~~ reward & the penalty levels change according to the SLA.
* Dotted line shows a quadratic approximation of the utility function.

1) Illustrate how control theory principles could be used for optimal resource allocation.

→ <u>Control Theory Principles</u> for optimal resource allocation

* Optimal control generates a sequence of control inputs over a look ahead horizon while estimating changes in operating ~~change~~ conditions.

* A convex cost funcⁿ has arguments $x(k)$, the state at step k, and $u(k)$, the control vector; this cost funcⁿ is minimized, subject to the constraints imposed by the system dynamics.

* The discrete-time optimal control problem is to determine the sequence of control variables $u(i), u(i+1) \ldots, u(n-1)$ to minimize the expression

$$J(i) = \phi(n, x(n)) + \sum_{k=i}^{n+} L^{k}(x(k) \cdot u(k))$$

$\phi(n, x(n)) \rightarrow$ cost funcⁿ of final step n.

$L^{k}(x(k) \cdot u(k)) \rightarrow$ time varying cost funcⁿ at intermediate steps

* The maximization is subject to the constraints.

$$x(k+1) = f^{k}(x(k) \cdot u(k))$$

$x(k+1) \rightarrow$ system state at time k+1

$u(k) \rightarrow$ input at time k.

$x(k) \rightarrow$ system state at time k

$f^{k} \rightarrow$ time-varying function (thus superscript).

(10) Illustrate Fair Queuing method

→ Fair queuing algorithm requires that separate queues, one per flow, be maintained by a switch and that the queues be serviced in Round-Robin manner.

↳ guarantees fairness of buffer space management.

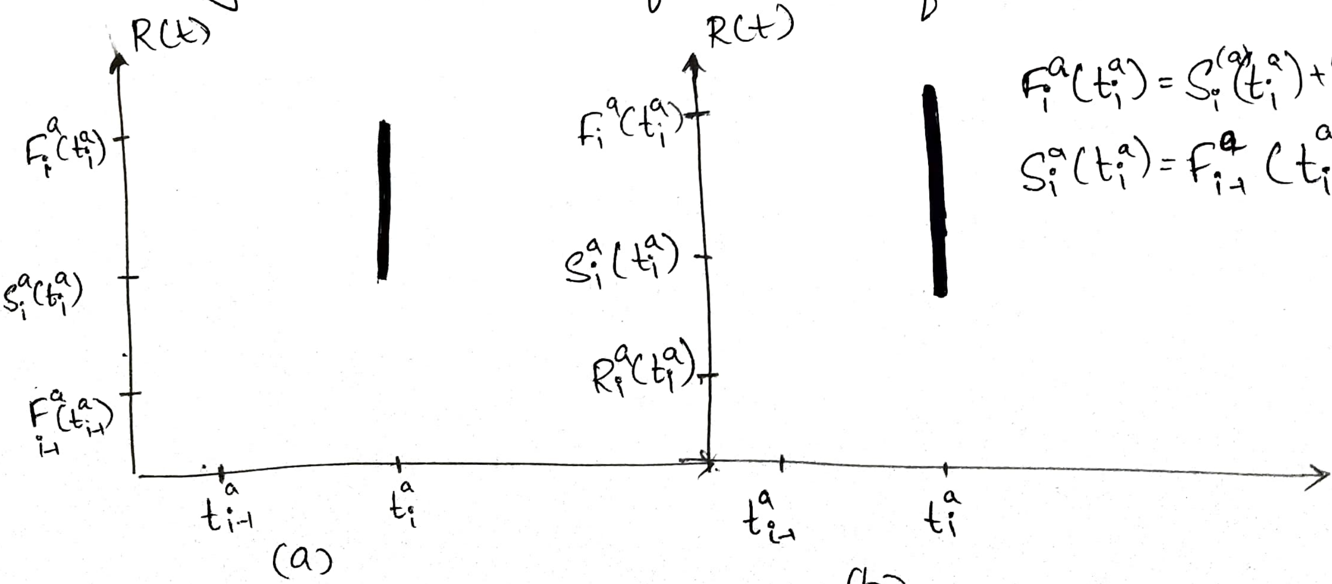
↳ disadv – doesn't guarantee fairness of bandwidth allocation.

→ Another solⁿ in fair queuing is by bit-by-bit round robbin strategy (BR).

↳ A single bit from each queue is transmitted & the queues are visited in a round robin fashion.

→ Fair allocation of bandwidth doesnot have an effect on the timing of the transmission.

↳ possible strategy is to allow less delay for the flows using less than their fair share of the bandwidth



$$F_i^a(t_i^a) = S_i^a(t_i^a) + P_i^a$$
$$S_i^a(t_i^a) = F_{i-1}^a(t_{i-1}^a)$$

The transmission of packet i of a flow can only start after the packet is available & the transmission of the previous packet is over.

(a) The new packet arrives after previous one is over.

(b) The new packet arrives before previous one was finished.