CHAPTER 1

A real time systems is ... ions"
required to complete its ... that takes
work and deliver its ... ain consideration
services on a timely ... ioes not play
basis. ... of jobs.

→ Example of Real-time Systems:-

1.) Digital Control.
2) Signal Processing.
3) command and control.
4) Telecommunication System

Some examples of RTS:

→ When we drive, they control the engine and brakes of our car and regulate traffic lights.

→ When we fly, they schedule and monitor the take off and landing of our plane, make it fly, maintain its flight path, and keep it out of harm's way.

→ When we are sick, they may monitor and regulate our blood pressure and heart beats.

→ when we are well, they can ~~entrain~~ entertain us

2070-07-21

CHAPTER 1

"Typical Real-Time Applications"

→ Real Time system:- The system that takes time as main consideration

→ In non real time system, time does not play critical role but usefulness of jobs.

→ Example of Real time Systems:-

1) Digital Control.
2) Signal Processing.
3) command and control.
4) Telecommunication System

Some examples of RTS:

→ When we drive, they control the engine and brakes of our car and regulate traffic lights.

→ When we fly, they schedule and monitor the take off and landing of our plane, make it fly, maintain its flight path, and keep it out of harm's way.

→ When we are sick, they may monitor and regulate our blood pressure and heart beats.

→ when we are well, they can ~~entrain~~ entertain us

with electronic games and joy rides.

Note:- In case of malfunctioning of some RTS, can have serious consequences. for This reason, a major emphasis is on technique for validating real-time systems.

→ By, validation, we mean a rigorous demostration that the system has the intended timing behavior.

B.V.S 2068

1. Digital controls:- Many real-time systems are embedded in sensors and actuators and function as digital controllers. As in fig, the term plant in the block diagram refers to a controlled system (for eg engine, brake, an aircraft). The state of the plant is monitored by the sensors and can be changed by actuators. The real time (computing) system estimates from the sensor readings the current state of the plant and computes a control output based on the

→ embedded
d functions
depicted
→ be dep.
tation that
d on the
stated
state
ed control-

difference between the current state and the desired state (called reference input in the figure). We call this computation the control-law computation of the controller. The output thus generated activates the actuators, which bring the plant closer to the desired state.

Note:- ...rides. ...RTS, can ...this reason, ...ue for vali- ...ous demos- ...ve intended

B.N.S 2068

1. Digital Control:-

→ Many real-time systems are embedded in sensors and actuators and functions as digital controllers.

→ The application of digital control system can be depicted ~~stood by~~ from following figure:-

Control-law Computation:- The computation that computes a control output based on the difference between the current state (given by sensor) and the desired state (called reference input) is called control-law computation.

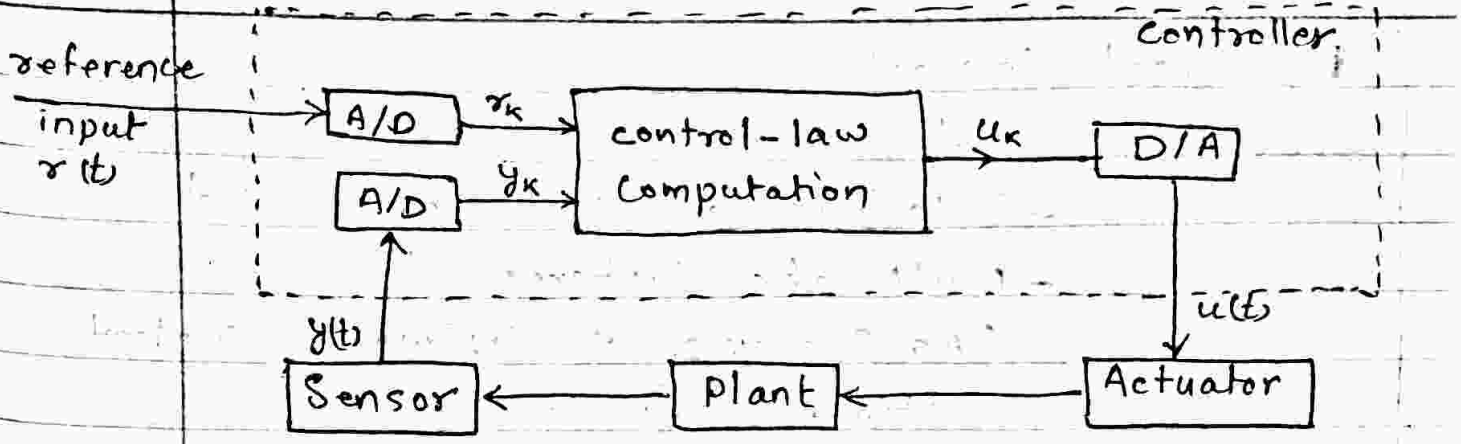


figure:- A digital controller

In diagram:-

A/D → Analog to digital converter

D/A → Digital to analog converter

r(t) → desired state (refference input)

y(t) → measured state (by sensor)

r_k, y_k → sampled values

Plant → is a controlled system for eg:- an engine, a brake, an aircraft, a patient etc.

Actuator → It activates the plant.

Sensor → It senses

$e(t) = r(t) - y(t)$ (is an error)

u(t) → output of the controller.

where u(t) consists of three terms:

$$u(t) \propto e(t)$$

$$u(t) \propto \int e(t)\,dt$$

$$u(t) \propto \frac{d\,e(t)}{d(t)}$$

1. Digital Control

1) Sampled Data System
   - Selection of sampling period
   - Multirate Systems.
     (An Example of Software control structures.)
   - Timing Characteristics.

2) More complex Control-Laws Computations.
   - Deadbeat Control
   - Kalman filter

1.1.1 Sampled Data Systems:- It is a common approach of designing a digital controller is to start with an analog controller that has the desired behavior in which analog version is transformed into a digital version. The resultant controller is a **Sampled Data System.**

→ It periodically samples analog signal reads (or samples) and digitizes the sensor reading periodically, carries out control-law computation every period and convert back digital outputs to analog form needed to activate the actuators.

A simple example: (PID)

An analog single-input / single-output PID (Proportional, Integral and Derivative) controller.

In the sampled data version,

- y_k, r_k for $k = 0, 1, 2, \ldots$ → input to control law computation.
- $e_k = r_k - y_k$ → k^{th} sample value of $e(t)$.
- There are many ways to discretize the derivative and integral of $e(t)$.
- for eg:- we can approximate the derivative of $e(t)$ for $(k-1)T \leq t \leq kT$ by $(e_k - e_{k-1}) / T$ and use the trapezoidal rule of numerical integration to transfor a continuous integral into a discrete form.
- The result is the following incremental expression of the k^{th} output u_k:

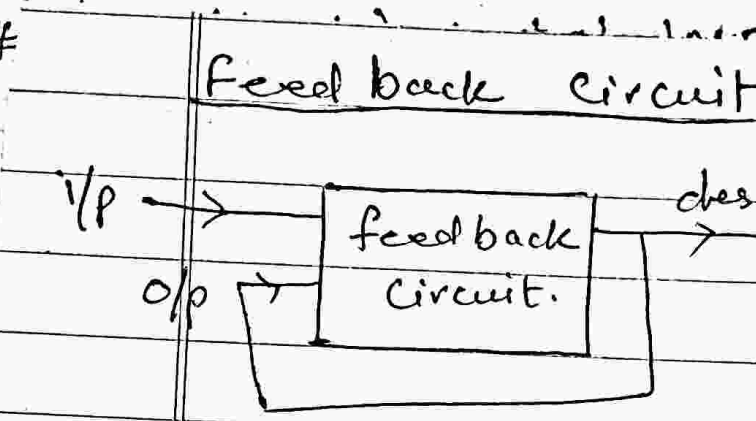
$$u_k = u_{k-2} + \alpha e_k + \beta e_{k-1} + \gamma e_{k-2} \quad \text{— (1)}$$

where, α, β, γ → proportional constants chosen at design time.

- During the k^{th} sampling period, the real time system computes the output of the controller according to eqn (1)
- This computation takes no more than 10-20 machine instructions -

- Different discretization methods may lead to different expressions of u_k but are simple to compute.

#



Trapezoidal rule of numerical integration.

, the k^{th}), the
the current
; for $i \leq k$. the
for $i > k$ in
system in
loop or simply

ller:-

riodically with

period T;
at each timer interrupt, do
do analog-to-digital conversion to get y;
compute control output u;
output u and do digital to analog conversion;
end do;

→ assumption:- → Timer provided by System.

may lead to
t are simple

(Feedback) control loop:-

During the sampling period (say the k^{th}), the control output u_k depends on the current and past measured values y_i for $i \leq k$. The future measured values y_i's for $i > k$ in turn depend on u_k. Such a system is called a (feedback) control loop or simply a loop.

Algorithm for digital controller:-

Set timer to interrupt periodically with period T;
at each timer interrupt, do
    do analog-to-digital conversion to get y;
    compute control output u;
    output u and do digital to analog conversion;
end do;

→ assumption:- → Timer provided by System.

→ the program setting generate interrupt for every T unit of time
→ stops generating interrupt when program setting cancelled.

Selection of Sampling Period:- ⇒ see last page for more

→ The length T of time between any two consecutive instants at which $y(t)$ and $r(t)$ are sampled is called the sampling period.
→ T is a key design choice. We want a sampling period T that achieves a good compromise. [it should not be too small in order to make system like analog version but small T cause frequent control law computation and higher processor time demand]
We need to consider
→ Two factors (for making selection of T) :

The first is the perceived responsive of overall system (i.e. plant & the controller). Oftentime, the system is operated by a person (eg, a driver or pilot). The operator may issue a command at any time, say at t. The consequent change in the reference input is read and reacted to by the controller at the next sampling instant. This instant can be as late as $t + T$. Thus sampling introduces a delay in the system response. The operator will feel the system sluggish when the delay

exceeds a tenth of a second. Therefore, the sampling period of any manual input should be under this limit.

→ The second factor is the dynamic behavior of the plant. We want to keep the oscillation in its response small and the system under control.

Read H/w

Illustration of this second factor → the arm of a disk drive → Reading as H/w.

Page 12

Multirate system:-

In case of multirate system, multiple inputs are given to obtain multiple outputs. Digital controller having more input with more output signals are multirate digital controller.

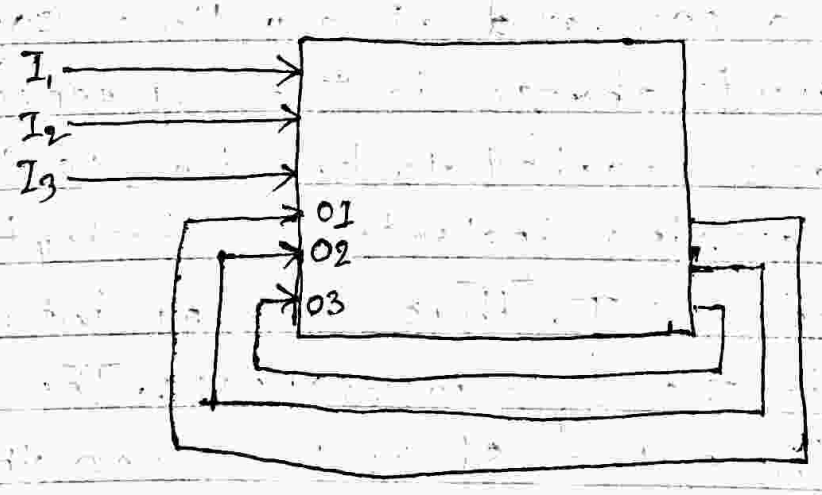


Fig:- Multirate System.

→ A plant typically has more than one degree of freedom. Its state is defined by multiple state variables (eg:- the rotation speed, temperature etc. of an engine or the tension and position of a video tape). Therefore, it is monitored by multiple sensors and controlled by multiple actuators.

→ Because different state variable may have different dynamics, the sampling periods required to achieve smooth responses from the perspective of different state variable may be different.

→ for eg: because the rotational speed of a engine changes faster than its temperature, the required sampling rate for RPM (Rotation per minute) control is higher than that for the temperature control.]

→ We can use the highest of all required sampling rates. This choice simplifies the controller software since all control laws are computed at the same repetition rate. However, some control-law computation are done more frequently than necessary; some processor time is wasted. To prevent this waste, multivariate

digital controllers usually use multiple rates and are therefore called multirate systems

→ The sampling periods in a multirate system are related in a harmonic way, that is, each longer sampling period is an integer multiple of every short period.

An Example of Software control structure.:- It is an example that shows the software structure of a flight controller.

→ The plant is a helicopter.

→ It has three velocity components, together they are called "collective".

→ Three rotational (angular) velocities, referred to as roll, pitch and yaw.

→ Uses three sampling rates 180, 90 & 30 Hz.

→ After initialization, the system executes a do loop at the rate of one iteration every $1/180$ second.

→ a cycle means a $1/180$-second cycle

→ the term computation means a control-law computation.

Algorithm for Software control structure of a flight controller.

Do the following in each 1/180-second cycle:

- Validate sensor data and select data source; in the presence of failures, reconfigure the system.
- Do the following 30-Hz avionics tasks, each once every six cycles.
  - Keyboard ~~and~~ input and mode selection.
  - data normalization and coordinate transformation.
  - tracking reference update.
- Do the following 30-Hz computations, each once every six cycle:
  - control laws of the outer pitch-control loop
  - control laws of the outer roll-control loop
  - control laws of the outer yaw- and collective-control loop
- Do each of the following 90-Hz computations once every two cycles, using output produced by 30-Hz computations and avionics tasks as input:
  - control laws of inner pitch-control loop.
  - control laws of inner roll and collective-controll loop.

- Computer the controll laws of the inner yaw-control loop, using outputs produced by 90-Hz control-law computation as input.
- Output commands.
- Carry out built-in-test.
- Wait until the begining of the next cycle.

Timing characteristics :-

→ Work-load generated by each multivariate, multirate digital controller consists of a few periodic control-laws computations. Their periods range from a few milliseconds to a few seconds.

→ A control system may contain numerous digital controllers, each of which deals with some attribute periodic of the plant. Together they demand tens or hundreds of control laws be computed periodically, some of them continuously and other only when requested by the operator or in reaction of some events.

→ The control laws of each multirate controller may have harmonic periods. They typically use the data produced by eachother as inputs and are called a rate group.

Each control-law computation can begin immediately after the begining of each sampling period when the most recent sensor data become available.

There is
→ No control theoretical reason to make sampling perrods of different rate groups related in a harmonic way.
→ It is natural to want the computation complete and hence, the sensor data processed before the data taken in the next period become available.
This objective is met when the response time of each control-law computation never exceeds the sampling period.

More Complex Control-Law Computation

Assumptions - for a PID or similar digital controller;

First
(1) Sensor data give accurate estimates of the state-variable values being monitored and control.
(not valid if noise or disturbances inside or out side the plant prevent accurate observations of its state)
(2) Second - The sensor data gives the state of the plant. In general, sensors moni

for some observable attributes of the plant. The values of the state variables must be digitized sensor readings.

3) Third → All the parameters representing the dynamics of the plant are known. (But this assumption is not valid for some plants. for eg:- is a flexible robot arm. Even the parameters of typical manipulators used in automated factories are not known accurately.)

⇒ When any of the simplifying assumptions is not valid, the simple feedback loop no longer suffices.
Since these assumptions are often not valid, the more complex ~~computation~~ control-law computations implemented as follows:-

Set timer to interrupt periodically with period T;
at each clock interrupt, do
sample and digitize sensor-readings to get measured values;

compute control output from measured and state-variable values;
convert control output ~~from measured~~ to analog form;
{ estimate and update plant parameters;
compute and update state variables;
end do;

→ The last two steps in the loop can increase the processor time demand of the controller significantly.

→ Two examples where the state update step is needed are:-
- Deadbeat control
- Kalman filter

Deadbeat Control

→ A discrete-time control scheme that has no continuous-time equivalence is deadbeat control.

→ In response to a step change in the reference input, a deadbeat controller brings the plant to the desired state by exerting or

the plant a fixed number (say n) of control commands.

→ A command is generated every T seconds (T is still called a sampling period).

→ Hence the plant reaches its desired state in nT second.

• → (In principle, the control-law computation of a deadbeat controller is also simple) the output produced by the controller during the kth sampling period is given by

$$u_k = \alpha \sum_{i=0}^{k} (r_i - y_i) + \sum_{i=0}^{k} \beta_i x_i$$

where, α, β_i → constant chosen at design time.

x_i → the value of the state variable in the i^{th} sampling period.

→ During each sampling period, the controller must compute an estimate of x_k from measured values y_i for $i \leq k$.

i.e. the state update ~~state~~ in the above do loop is needed.

Kalman filter

→ Kalman filtering is a commonly used means to improve the accuracy of measurements and to estimate model parameters in the presence of noise and uncertainty.

Example:- To illustrate

* A Simple monitor system *

y_k → measured value at k^{th} sampling period.

x_k → estimated value of a state variable.

x → a constant value of a state variable at starting time 0.

$y_k = x + \varepsilon_k$; the measured value due to noise.

$\varepsilon_k \rightarrow 0$: average value of random variable.

$\varepsilon_k \rightarrow \sigma_k$; standard deviation of " "

Kalman filter starts with Initial estimate $\hat{x}_1 = y_1$

and compute a new estimate each sampling period.

Specifically, for $K \geq 1$, the filter computes —

$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - \hat{x}_{k-1}) \quad \text{——(1.2a)}$$

where,

$$K_k = \frac{P_k}{\sigma_k^2 + P_k} \quad \text{——(1.2b)}$$

Where,

K_K → Kalman gain

P_K → the variance of estimation error $\tilde{x}_K - x$

$$P_K = E[(\tilde{x}_K - x)^2] = (1 - K_{K-1}) P_{K-1} \quad \text{(1.2c)}$$

(E: expected value of)

P_K decreases with increasing K

P_K at steady-state variance, for large K.

In a multivariate system,

$$y_K = A x_K + e_K$$

y_K → is an n-dimensional vector.

e_K → vector, additive noise

where A → $n \times n'$ measurement matrix

→ n'-dimensional vector, if during each sampling period, the reading of n' sensors are taken.

Now, Eqn (1.29) becomes an n-dimensional vector eqn:-

$$\tilde{x}_K = \tilde{x}_{K-1} + K_K (y_K - A\tilde{x}_{K-1})$$

Similarly,

Kalman gain K_K and variance P_K are given by the matrix version of eqn (1.2b) and (1.2c). So, the computation in

each sampling period involves a few matrix multiplications and additions and one matrix inversion.

High-Level Controls

→ Controllers in a complex monitor and control system are typically organized hierarchically. One or more digital controllers at the lowest level directly control the physical plant.

→ Each output of a higher level controller is a reference input of one or more lower-level controllers.

→ Mostly, the higher-level controller interfaces with the operator(s).

For eg:- - A patient care system.
- An Air traffic / flight control hierarchy.
- control system of robot in a factory.

→ In an air traffic control system. The ATC is a high level controller which controls the flight management. The aeroplane and its control system within the plane is lower level controllers which helps

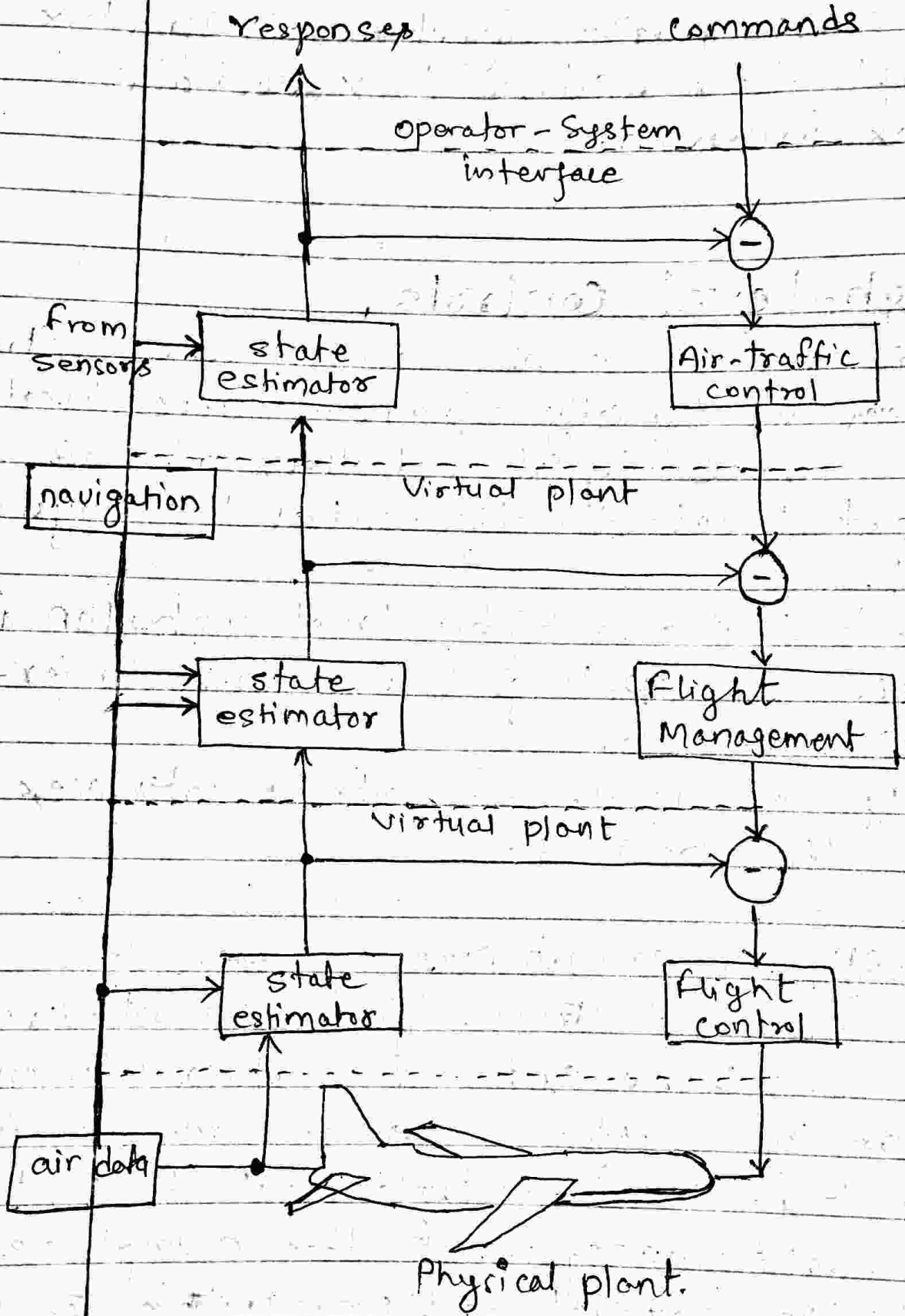


figure:- Air traffic / flight control hierarchy.

maintaining altitude, direction, speed etc and the pilot is the operator.

(~~Real time command~~)

2.2.2 Guidance and control :-

→ While digital control deals with the dynamic behaviour of physical plant, a second level controller typically performs guidance and control to achieve the optimum goal of lower level controller.

→ The trajectories is most desirable since it optimizes some cost function. The algorithm used for this purpose is the solution of some constrained optimization problem.

→ For eg:- In flight management system, the constraints for chosen flight path include the maximum and minimum allowed cruise speeds, decent / accent rates, as well as constraints imposed by external factors like the ground track and altitude profile specified by ATC system and weather conditions.

→ A cost function is fuel consumption : A most desirable flight path is a most fuel

efficient among all paths that meet all the constraints and will bring the aircraft to the next metering fix at the assigned ~~time~~. arrival time. This problem is known as the constrained fixed-time, minimum-fuel problem.

→ When the flight is late, the flight management system may try to bring the aircraft to the next metering fix in the shortest time. In this case, it will use an algorithm that solves the time-optimal problem.

N.P

Complexity and Timing requirement:-

→ The optimization problems for a guidance (or path planning) system are typically nonlinear.

→ Those problems can be solved by using dynamic programming and mathematical programming techniques.

→ But most of the optimal algorithms do not guarantee ~~to~~ to find a usable solution.

→ Heuristic algorithms for guidance and control purpose typically uses one constraint at a time. And they start with

an initial condition and some initial solution and adjust the value of one solution parameter at a time until a satisfactory solution is found.

→ Unlike digital controller, a guidance system does not need to compute its control laws frequently.
→ So it can be done off-line and are not time-critical.
→ While in-flight, the system still needs to compute some control laws to monitor and control the transitions between different flight phases. as well as algorithms for estimating and predicting time to waypoints and so forth.
→ These time critical computations tend to be simpler and more deterministic and have periods in order of seconds and minutes.
→ When the precomputed flight plan needs to be updated or a new one computed in-flight, the system has minutes to compute and can accept suboptimal solutions when there is no time.

Not done

Other capabilities:-

→ The higher level control system often interfaces with the operator and other systems.

→ To interact with the operator, it updates displays and reacts to operator commands.

→ Other systems for examples are:- a voice, telemetry, or multimedia communication system that supports operator interactions. Other examples are radar and navigation devices.

→ The control system may use the information provided by these devices and partially control these devices.

→ An avionic or flight management system ~~to~~ update the display. ~~The updates~~ of radar, flight path, and air-data information.

→ Similarly, it periodically updates navigation data provided by inertial and navigation aids.

→ An avionics system of a military aircraft also does tracking and ballistic computations and coordinates radar and coordinate radar weapon control systems, and it does them with repetition periods of a few to a few hundred milliseconds.

1.2.3 Real-Time command and control:-

→ The controller at the highest level of control hierarchy is called a real-time command and control system. for eg: Air Traffic control (ATC) system.

[→ The ATC system is at the highest level. It regulates the flow of flights to each destination airport by assigning to each aircraft an arrival time at each metering fix en route to the destination.]

→ The low level controller work load is purely or mostly periodic. The command and control computes and communicates in response to sporadic events and operators command. It may process speech and image, query and update database, simulate various scenarios etc. The resource and processing time would be large. The command & control is generally a large distributed system containing many computers and different networks. It is similar to interactive online transaction processing system.

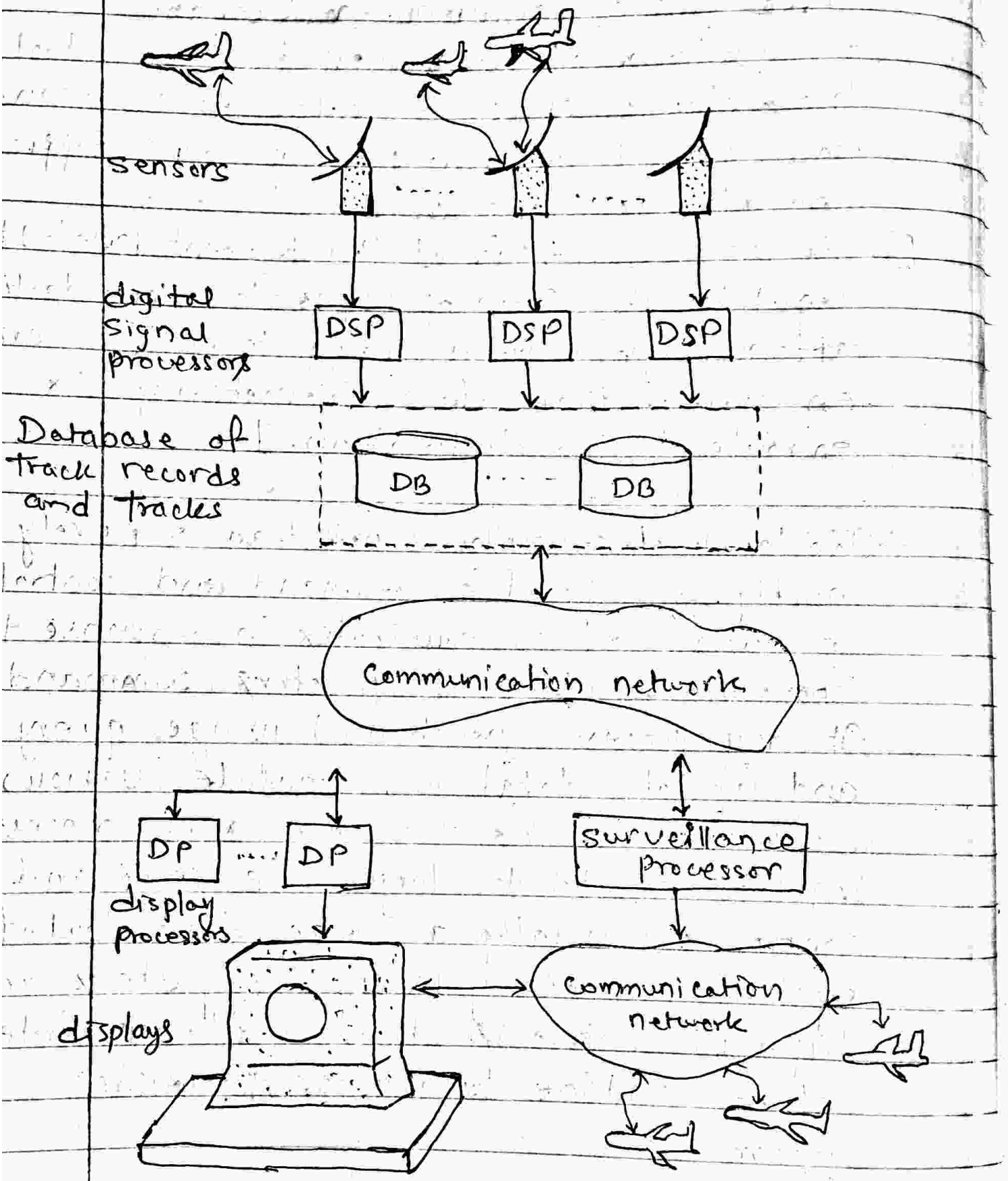


figure:- An architecture of air traffic control system

An Air Traffic control (ATC) system :-

→ The ATC system monitors the aircraft in its coverage area and ~~the~~ the environment (eg:- weather condition) and generates and presents the information needed by the operators (i.e, the air traffic controller). Outputs from the ATC system include the assigned arrival time to metering fixs for individual aircraft. Those outputs are inputs to on-board flight management systems. Thus the ATC system indirectly controls the embedded components in low levels of the control hierarchy. In addition, the ATC system provides voice and telemetry links to on-board ~~avoin~~ avionics. Thus it supports the communication among the operators at both levels. (i.e. pilot and air traffic controllers).

The ATC system gathers information on the "state" of each aircraft via one or more active radars. Such a radar interrogates each aircraft periodically. When interrogated, an aircraft responds by sending to the ATC system its "state variables": identifier, position, altitude, heading and so on. Those variables are collectively reffered to as track. record and the current trajectory of the ~~tracks~~

aircraft is a track. The ATC system processes messages from aircraft and stores the state information thus obtained in a database. This information is picked up and processed by display processors. At the same time, surveilance system continuously analyzes the senario and ~~alert~~ alerts the operators whenever it detects any potential hazard (eg a possible collision). Again the rates at which human interfaces (eg keyboard and displays) operate must be at least 10 Hz. The other response times can be considerably larger. for eg:- the allowed response time from radar inputs is one to two seconds, and the period of weather updates is in the order of ten seconds.

13 Signal Processing :- Any thing that carries information is called signal. It is a real or complex function of one or more variables. for eg temperature is a one dimentional signal, physical system can be three dimensional or four dimensional

Processing :- Processing means operation in some fashion on a signal to extract some useful information. The signal is processed by a system and the signal processor can be electronic, mechanical or a program.

Signal processing may include digital filtering, audio, video and image compression or decompression, radar signal processing etc

Processing × Bandwidth × Demand :-

The bandwidth demand of a signal processor depends on the sampling rate and the no. of outputs at a time. The real time signal processing applications compute one or more outputs in each sampling period. Each output $x(k)$ is a weighted sum of n inputs $y(i)$'s

$$\text{i.e. } x(k) = \sum_{i=1}^{n} a(k,i)\, y(i) \quad \text{---} \quad (1)$$

when $a(k,i)$ are the weights whose value is known and fixed. The weight value in a signal processing system takes at least one addition and multiplication. The equation (1) means the given representation of an object is transformed into another representation in terms

of output. The processor output demand of an application also depends on the no. of outputs. It is required in each sampling period. for eg in a noise filtering the sampling rate is few kiloherz to ten's of kiloherz. Therefore, such application perform $10^4 - 10^7$ multiplications and additions per second. The time required to produce output is $O(n)$. In a filtering system the value of n ranges from 10's to 100. The no. of outputs may be of the (order n). $O(n)$ and the complexity of the computation increases by $O(n^2)$ such as in image processing in which the space representation of each image is transformed into another form. for example, consider a mxm 30 frames/sec video. Therefore, the no. of inputs $n = m^2$. The transformation of each frame takes m^4 multiplication & addition per second. If $m = 100$ pixels then the transformation of video takes 3×10^9 additions and multiplication per second.

[Note $30 \times (m^4) = 30 \times (100^4) = 3 \times 10^9$]

(S.Q) Q.No. 6
2068

Radar System

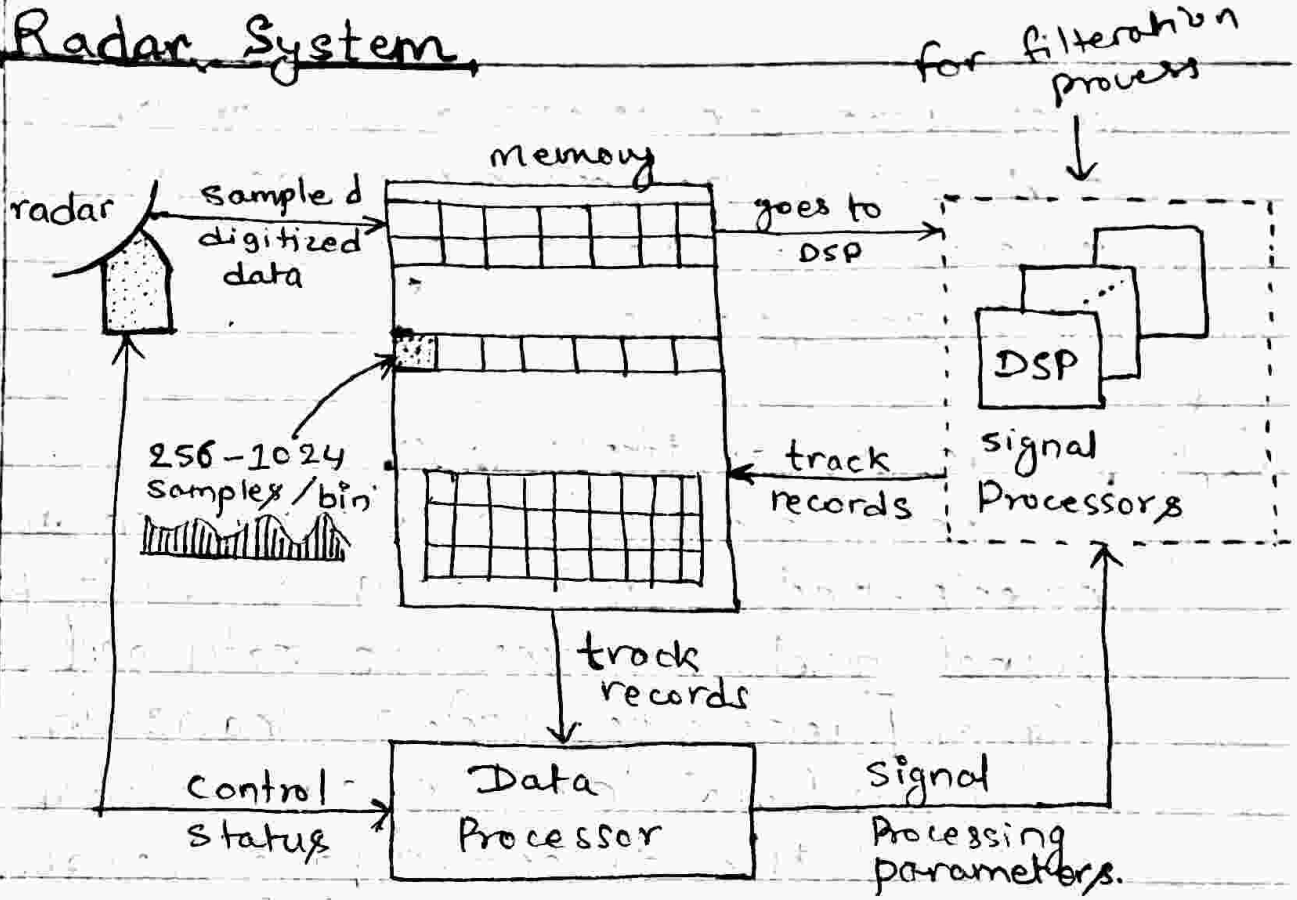


Figure: Radar Signal processing and tracking system.

A signal processing application is typically a part of a larger system. As an example, Figure shows a block diagram of radar signal processing and tracking system. The system consists of input output subsystem that samples and digitize from the radar and places the sampled values in a stored memory. An array of the echo signal processors processes these digital signal sampled values. The data thus produced are analyzed by one or more data processor, commands to control the radar and select parameters which not only interface with the display systems, but

also generate commands to control the radar and select parameters to be used by signal processors in the next cycle of data collection and analysis.

Q.N. 6 2068

Radar Signal Processing

The radar are used to find out the moving objects. Radar first sends a short frequency signal and captures the returned echo signal. [Generally, radar scans in all direction (360°) and every return signal from all the direction] are recorded & stored in radar memory. These stored signals are processed by digital signal processor. And a track record is generated. The track record gives the trajectory of the moving objects, its position & velocity. The track record is provided to a data processor which analyzes the record and generates control signals for the radar and DSPs (digital signal processors). The radar memory is divided into small partition called Bin. The bin is equivalent to a small distance during which the samples are generated.

perishable → distroyable, dynamically change होईशकते।

Trackers:-

- Gating.
- Data Association.

→ What is false return?
A track record on a nonexisting object is called a false return.

→ Real time databases are perishable, why?
Time is crucial in real time dabase, we track the data time to time. Since, time plays very important role and it updates continuously. it is perishable in nature.

→ A trade record on a non-existing object is called a false return. An application that examines all the track records in order to sort out false returns from real ones and update the trajectories of detected objects is called a tracker.

→ If the trajectory is an existing one, the measured value assigned to it gives the current position and velocity of the object moving along the trajectory. If the trajectory is new, the measured value gives the position and velocity of a possible new objects. In

The example in figure, the tracker runs on one or more data processors which communicate with the signal processors via the shared memory.

→ The tracking can be performed using two methods

i) Gating
ii) Data Association

i) Gating :-

Gating is the process of putting each measured value into one of two categories depending on whether it can or cannot be tentatively assigned to one or more established trajectories. The gating process tentatively assigns a measured value to an established trajectory if it is within a threshold distance G away from the predicted current position and velocity of the object moving along the trajectory.

→ The threshold G is called the track gate.

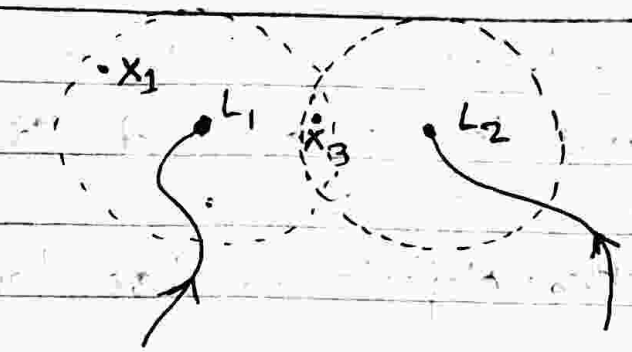


Figure :- Gating Process

In figure: L_1 & L_2 are two established trajectories.

x_1, x_2 and x_3 are the measured values given by three track records.

x_1 is assigned to L_1 because it is within G- from L_1.

x_3 is assigned to both L_1 and L_2.

But x_2 is not assigned to any of the trajectories. It represents either a false return or a new object. Since it is not possible to distinguish between these two cases, the tracker hypothesizes that x_2 is the position of a new object.

ii) Data Association :- (Data Acquisition)

→ The tracking process completes if, after every gating, every measured value is assigned to at most one trajectory and every trajectory is assigned to at most one measured value.

- This can occur when
i) The radar signal is strong and the interferences is low.
ii) The density of the object is also low.

→ But in many cases, some measured values can be assigned to more than one trajectories or a trajectory can be assigned to more than one measured value. In such cases many data association algorithms can be applied.

→ Nearest Neighbouring algorithm:-
It works as follows:-

1) Examine the ~~gating~~ tentative assignments produced by the gating step.
a) for each trajectory that is tentatively assigned a single unique measured value, assign the measured value to the trajectory. Discard from further examination the trajectory and the measured value, together with all tentative assignments involving them.
b) for each measured value that is tentatively assigned to a single trajectory, discard

The tentative assignments of those measured values that are tentatively assigned to this trajectory if the values are also assigned to some other trajectories.

2) Sort the remaining tentative assignments in order of nondecreasing distance.
3) Assign the measured value given by the first tentative assignment in the list to the corresponding trajectory and discard the measured value and trajectory.
4) Repeat step (3) until the step of tentative assignments is empty.

→ In data association step, the ambiguities should be removed. In the figure, x_3 is in between L_1 & L_2. Similarly x_2 is in outside of these classes. These things should be handled seriously to obtain accurate records.
After applying data association step in above diagram; x_2 is removed from the memory & x_3 is kept in L_2.

Complexity and Timing Requirements:-

→ The amounts of processor time and memory space required by the tracker are data dependent and can vary widely.

→ When there are n tractories and m measured values, the time complexity of gating is $O(nm \log m)$.

→ In the worst case, the time complexity of gating is $O(nm \log nm)$. (by NNA algorithm)

→ The amount of time and space required by multiple-hypothesis tracking grow exponentially with the maximum number of hypothesis.

Other Real-time Applications:

→ Real-time database
→ multimedia applications.

(1) Real-Time Databases:-

The term real-time database systems refers to a diverse spectrum of information systems, ranging from stock price quotation systems to track records databases, to real-time

file systems. The data in Real-time database are perishable in nature.

→ Specifically, a real-time database contains data objects, called image objects, that represents real-word objects.

→ For eg:- An air traffic control database contains image objects that represents aircraft in the coverage area. [The value of these attributes are updated periodically based on the measured values of the actual position and heading provided by the radar system.] ×

The attributes of such an image object include the position and heading of the aircraft.

Age of image object:-

It's a duration of time when image objects exists.

a) Absolute temporal consistency:

Syntax:

Age ≤ Threshold (provided)

→ A set of data objects is said to be absolutely temporal consistent if the maximum age of the objects in the set is no greater than a certain threshold.

→ for eg:- If the threshold age is 5 second, the age of object can't be greater than 10sec so as to have absolute temporal consistent property.

b) Relative Temporal consistency:-

→ for relative temporal consistency, we need two objects whose maximum difference of age should not be larger than the threshold value. to get the relative temporal consistency.

for eg:- threshold = 5 sec

Age of one object = 10 sec.

Age of another object = 20 sec.

Hence, difference = 20 - 10 = 10 sec > 5 sec

So failed.

It should be less than 5 sec to have relative temporal consistency.

consistency model → Read from book

(2) Multimedia applications:-

→ Multimedia is one of the frequently encountered real-time applications.

→ A multimedia application may process, store, transmit, and display any number of

Moving Pictures experts group → MPEG

video streams, audio stream, images, graphics and text.

→ A video stream is a sequence of data frames which encodes a video. An audio stream encodes a voice, sound or music.

→ Without compression, the storage space and transmission bandwidth required by a video are enormous. Therefore, a video stream, as well as the associated audio stream should be compressed as soon as it is captured.

For eg:- MPEG, Compression / Decompression

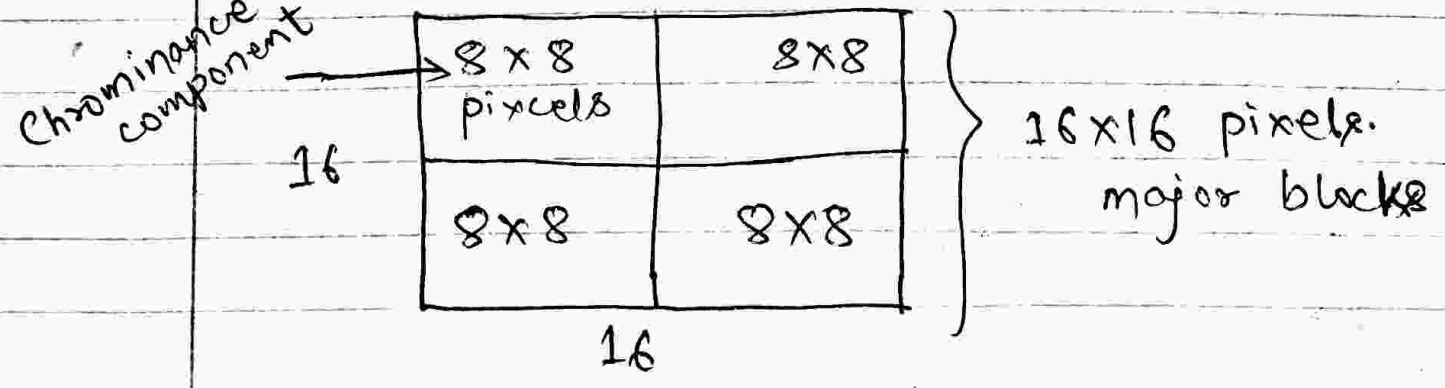
for this we use following steps:-

(i) Motion Estimation

(ii) DCT (Discrete Cosine Transform) and Encoding.

(iii) Inverse Transformation (Decompression)

In motion estimation, whole image is divided into multiple blocks. for eg:- 8x8 pixels



→ After blocking phase, the digital image is compressed using DCT equations.
→ And finally to decompress the compressed object, we apply inverse transformation rule.

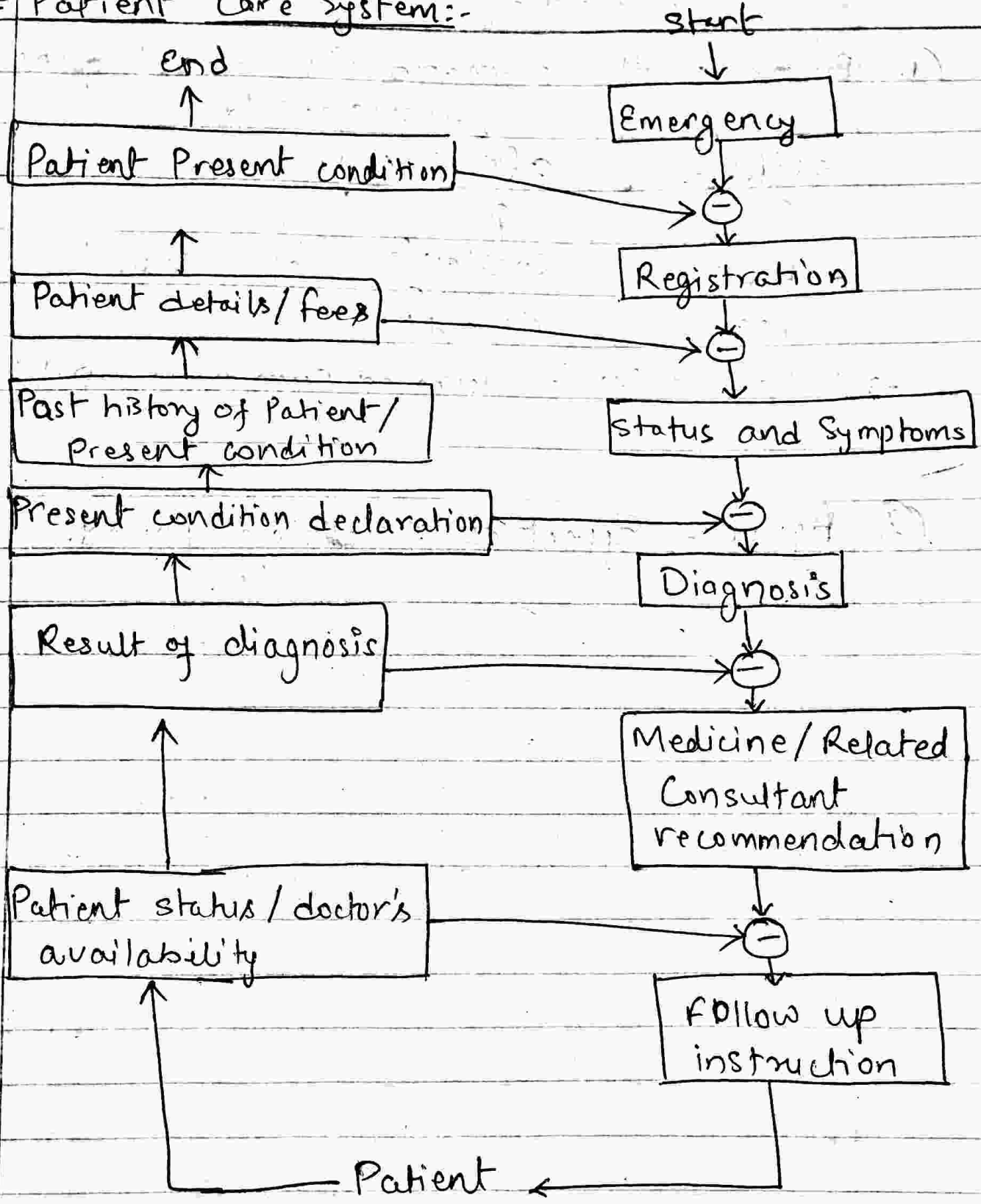
Real time characteristics → Read from Book.

(Joint photographic Experts Group)
→ JPEG for compression of still images.
→ MPEG for " " motion vedio.
→ CCITTI for " video telephony and telecon-freneing.

GIF → Graphics Interchange format.
PNG → Portable network Graphics format.

H/W

Patient Care System:-



Remained topic:-

① **Band width demand** (In signal processing)

$$x(k) = \sum_{i=1}^{n} a(k,i)\, y(i)$$

(known and fix)

- $x(k)$ ← output signal
- $a(k,i)$ ← constant value (weight)
- $y(i)$ ← input signal

→ Output $x(k)$ is weighted sum of n inputs $y(i)$.

② figure:- Effect of sampling period.

Chapter 2
2070-08-13

Hard Vs. Soft Real-Time System (4 Hrs)

→ Jobs and Processors
→ Release times
→ Deadlines and Timing constraints
→ Hard and soft Timing constraints.
→ Hard Real time System
→ Soft Real time System.

Jobs and processors:-

→ **Jobs:-** Each unit of work that is scheduled and executed by the system is a job and a set of ~~jobs~~ related jobs which jointly provide some system function is a task.

for example :- (Job)
- the computation of a control law is a job.
- The " of FFT (Fast Fourier Transform) " "
- The Transformation of a data packet.
- The retrieval of a file.

को सट्टा

[Note:- compute and transmit → job execute or is executed.
प्रयोग गर्दा]

Processors :- Every jobs execute on some resources like CPU, networks or disk. These resources are called servers in queuing theory literature. But they are called active resources in real-time systems literatures. In order to avoid overloading of the term server, we call these resources as processors.

Release Times, Deadlines and Timing Constraints

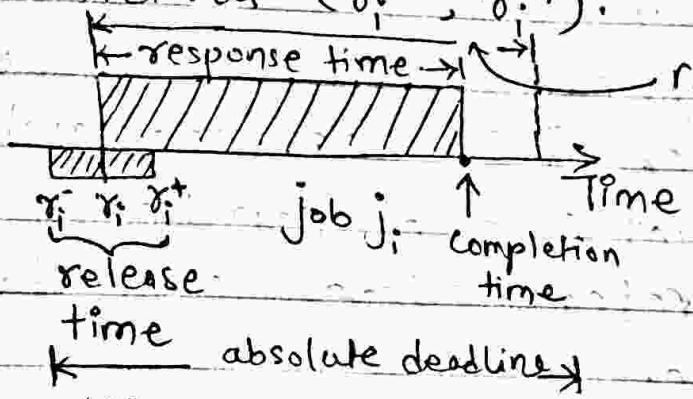
[Release time and Dead lines of jobs are two parameter that distinguishes job in real-time system from those in non-real-time systems.]

Release Time :-

→ The Release time of a job is the instant of time at which the job becomes available for execution.
The job can be scheduled and executed at any time at or after its release time whenever its {data and control dependency} conditions are met.
↓
resource dependency

→ Release time may jitter so it is indicated as

an interval (r_i^-, r_i^+).



Deadline :-

The deadline of a job is the instant of time by which its execution is required to be completed.

A job has no deadline if its deadline is at infinity.

The deadline of a job can be described in terms of a response time, relative deadline or absolute deadline.

→ Response time (completion time) :- The length of time from the release time of job to the instant when it completes.

types of deadlines

→ Relative deadline :- The maximum allowable response time of a job is called its relative deadline.

→ Absolute deadline :- The deadline of a job when equal to release time plus its relative deadline, is called absolute deadline.

Timing Constraint :-

constraint imposed on the timing behaviour of a job is called Timing constraint.

Timing constraint of a job can be specified in terms of its release time and relative or absolute time.

Some complex timing constraints can not be specified conveniently in terms of release times and deadlines.

Example of a Release time / Response time / Deadline

Consider a system used to monitor and control several heating furnaces. The system takes 20 ms to initialize when turned on. After initialization, every 100 ms ~~hundred millisecond~~, the system samples and reads the temperature sensor.

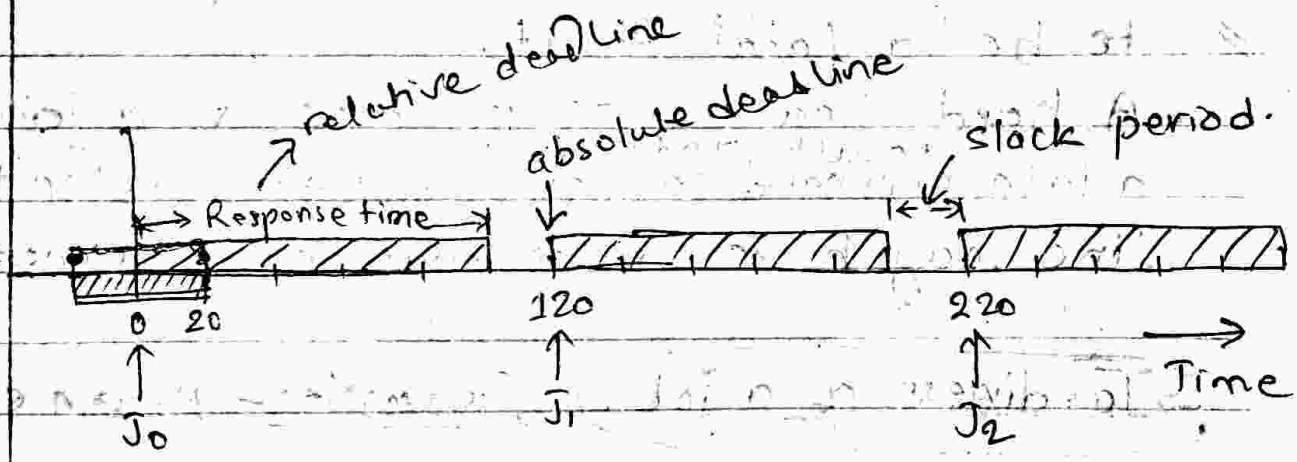
It computes the control laws of each furnace every 100 ms to process the temperature readings and determine flow of rates of fuel, air and coolant

→ Compute the control law for the furnace to process temperature readings, determined

the correct flow rate of fuel, air and coolant.

→ Adjust the flow rate to match the computed values.

Find out the released time, relative and absolute deadline and response time for the job $J_0, J_1, \ldots J_k$ of the system task.



The release time of any job for the given example is $(20 + K * 100)$ ms

for $J_0 = (20 + 0 * 100)$ ms $= 20$ ms

Similarly, $J_1 = (20 + 1 * 100)$ ms $= 120$ ms.

The relative deadline of the jobs is equal to 100 ms.

Absolute deadline (first job) $= (20 + (k+1) * 100)$ ms

∴ The absolute deadline of 1st job is

$= (20 + (0+1) * 100) = 120$ ms.

Similarly for $J_1 = 220$

$J_3 = 320$ and so on.

Hard and Soft Timing constraints

Hard (a) → commonly, a timing constraint or deadline is hard if the failure to meet it, is considered to be a fatal fault.

- A hard deadline is imposed on a job because a late result produced by the job after the deadline may have disastrous consequences.

→ Tardiness of a job:- Example:- As an example, a late command to stop a train may cause a collision, and a bomb dropped too late may hit a civilian population instead of intended military target.

soft → A timing constraint or deadline is soft if the failure to meet it, doesnot cause serious harm. A soft deadline is undesirable in Real time system. It causes the system's

overall performance poorer and poorer when more and more jobs with soft deadlines complete late.

(b) # Tardiness of a job :-

Hard and soft timing constraints can be defined as a function of (in the sense of) tardiness of a job in real-time literature.

→ The tardiness of a job measures how late it completes respective to its deadline.

- tardiness is zero if the job completes at or before its deadline; otherwise, if the job is late, its tardiness is equal to the difference between its completion time and its deadline.

→ • The usefulness of result produced by soft real-time job decreases gradually as the tardiness of the job increases.

- But the usefulness of the result produced by a hard real-time job falls off abruptly and may even become negative when the tardiness of the job becomes larger than zero.

2070-08-17

Unit Test 1

F.M. 20
P.M. 12

Attempt all.

(1) Define RTS. Describe Digital controller with neat diagram. (1+4)

(2) ~~Why the concept of~~ what is control law computation? Why the concept of complex control law computation has been emerged? Explain relating with assumptions. (2+3)

(3) Define radar signal processing. what is false return? (4+1)

(4) Define High-level control with Air Traffic control system. what is Real Time database (3+2)

what are the assumptions of simple PID controller and what is complex control law computation (2+1)

Another view of hard and soft Timing constraints :-

Timing constraints based on
- functional criticality of jobs,
- Usefulness of late result.
- deterministic or probabilistic nature of the constraints.

(a) functional criticality of jobs :-

→ timing constraint is hard if the failure to meet the deadline causes a fatal fault.

→ timing constraint is soft if the late result produced by the job after dead line may be considerable.

(b) tardiness of job :-

⇒ previous topic.

(c) Deterministic or probabilistic nature of constraints :-

→ If a job must never missed its dead line, then the deadline is hard.

→ If ~~a job never misses its~~ its deadline can be missed with some acceptably low probability, then its timing constraint is soft.

Hard Timing constraints and Temporal quality of service Guarantees

→ The timing constraint of a job is hard, and the job is a hard real-time job, if the user requires the validation that the system always meet the timing constraint.

→ Validation means the demonstration by a provably correct, effecient procedure or by exhaustive simulation and testing.

→ If no validation is required or only a demonstration that the job meet some statistical constraint suffices, then the timing constraint of the job is soft.

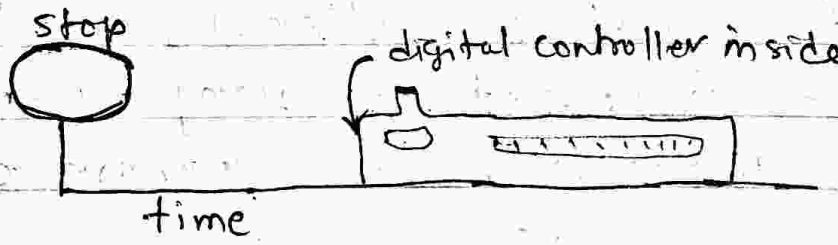
→ Stated another way, if the user wants the temporal quality (eg. response time and jitter) of the service provided by a task guaranteed and the satisfaction of the timing constraints defining the temporal quality validated, then the timing constraints are

hard.
On the other hand, if the user demands the best quality of service the system can provide but allows the system to deliver qualities below what is defined by the timing constraints, then the timing constraints are soft.

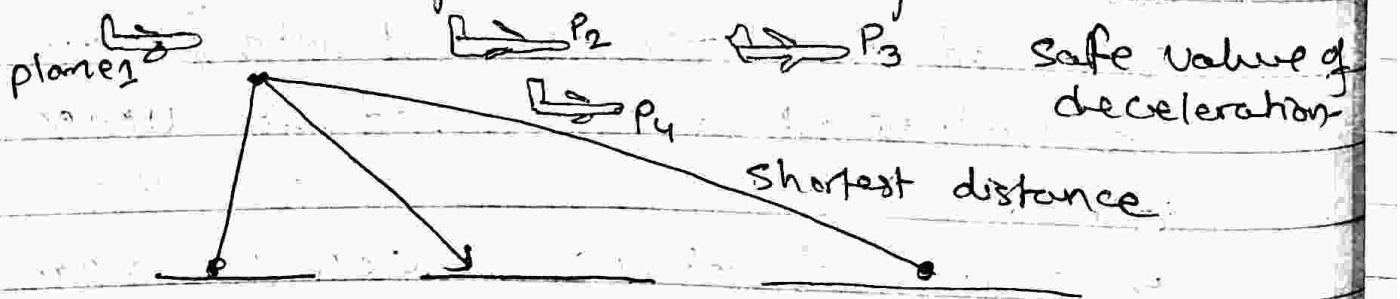
→ An application (task) with hard timing constraints is a hard real-time application. And a system containing mostly hard real-time applications is a hard real-time system.
[for many traditional hard real-time applications (eg. digital controllers), all the tasks and jobs executed in every operation mode of the system are known a priori.]

Hard Real time systems:- (Example)
Imposed hard timing constraints is required to be validated and guarded for users satisfaction.
for eg:- To illustrate reason for requiring timing guarantees, we consider an automatic controlled train. It cannot stop instantaneously. When the ~~train~~ signal is red

(stop), its braking action must be activated a certain distance away from the signal post at which the train must stop. This braking distance depends on the speed of the train and the safe value of decelaration. from the speed and safe value deceleration of the Train, the controller can compute the time for the train to travel the braking distance. This time in Turn imposes a constraint on the response time of the jobs which sense and process the stop signal and activate the brake.



other example :- ATC system.



Soft Real time Systems:- (example)

A system in which jobs have soft deadlines is a soft real-time system.

Example of such systems:-

- On line transaction system.
- telephone switches. (mobile n/w)
- electronic games ~~etc.~~
- Stock price quotation system etc.
- Multimedia Systems.

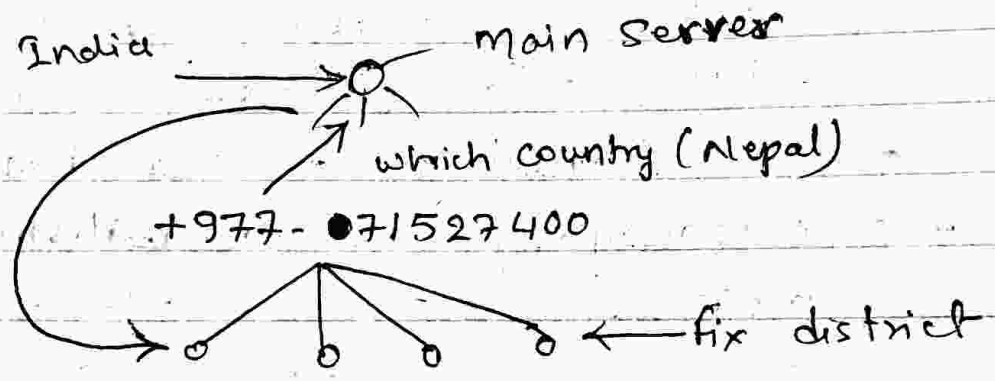
→ The Timing requirement of soft real-time systems are often specified in probabilistic terms. for example:→

(i) Telephone network :-

In response to our dialing a telephone number, a sequence of jobs in turn, each routes the control signal from one switch to another in order to set up a connection through the network on our behalf. We expect that our call will be put through in a short time. ~~To ensure it~~ This expectation is met most of the time by the network, a timing constraint may be imposed on this sequence of jobs as a design objective (eg. the sequence must complete in no more than 10 seconds

for 95 percent of the time and in no more than 20 seconds for 99.95 % of the time). The users are usually satisfied if after extensive simulation and trial use, the system indeed appears to meet this requirement.

(iv) Multimedia System :-



(ii) Multimedia System :- [Let us consider a for example :- multimedia system that provide ~~consider~~ the user with services of "guaranteed" quality.]

A frame of a movie must be delivered every (30th) ~~of~~ thirtieth of a second, and the difference in the times when each video frame is displayed and when the accompanied speech is presented should be no more than 80 msec.

In fact, it is common to subject each new video stream to be transmitted by a network to an acceptance test. If the network cannot guarantee the satisfaction of timing constraints of the stream without violating the constraints of existing streams, the new stream is rejected, and its admission is requested again at some later time. However, the users are often willing to tolerate a few glitches as long as the glitches occur rarely and for short lengths of time. At the same time, they are not willing to pay the cost of eliminating the glitches completely. for this ~~region~~ reason, we often see timing constraints of multimedia system guaranteed on a statistical basis.

2070-8-25

Chapter 3

A Reference Model of Real-Time systems

In a reference model of real-time systems, each system is characterized by three elements:

1) a workload model that describes the applications supported by the system.
2) A resource model that describes the system resources available to the applications, and
3) Algorithms that define how the application system uses the resources at all times.

→ first two elements of the reference model include description of the application & resources.
→ Third element is the set of scheduling and resource management algorithm.

Processors and Resources:-

All the system resources are divided into two major types: processors and resources.

→ Processors are called servers and active resources. for eg:- computer, transmission links, disks and database server etc.
They carry out machine instructions, move

data from one place to another, retrieve files, process queries and so on. Every job must have one or more processors in order to execute and make progress toward completion.

→ Two processors are of same type if they are functionally identical and can be used interchangeably.
for eg:- Symmetrical Multiprocessor (SMP) system.

→ Processors that are functionally different, or for some other reason can not be used interchangeably, are of different types, as they are functionally different.
for example:-
A transmission link connecting an on-board flight management system to the ground controller is a different type of processor from the link connecting two air traffic control centers even when the links have the same characteristics, because they cannot be used interchangeably.

We use 'P' to represent processor. The processors are $P_1, P_2, \ldots P_n$.

Resource :-

- Resources are passive resources. for eg:- memory, sequence numbers, mutexes, and database locks.
- A process job may need some resources in addition to the processor in order to make progress.

for example:-
A computation job may share data with other computations, and the data may be guarded by semaphores. we model each semaphor as a resource. ~~when~~

→ When a job wants to access the shared data guarded by a semaphore R, it must first lock the semaphore, and then it enters the critical section of the code where it access the shared data. In this case, we say that the job requires the resource R for the duration of this critical section.

→ We will use letter 'R' to denote resource. The resources in the example mentioned above are reusable, because they are not consumed ~~by another process~~ during use.

A confused or disordered state or collection

- To prevent model from being cluttered by irrelevant details, we typically omit the resources that are plentiful.

↑ existing in great quantity

→ A resource is plentiful if no jobs is ever prevented from execution by the lack of this resource.

Temporal Parameters of Real-time workload.

The workload on processors consists of jobs, ~~(set of tasks)~~ each of which is a unit of work to be allocated processor time and other resources.

Many paramenters of hard real time jobs and tasks are known at all time, otherwise it wouldn't be possible to ensure that the system meet its requirements.

Each job J_i is characterized by its temporal parameters, functional parameters and interconnection parameter.

Temporal parameters tells us its timing constraints and behaviour. functional parameter

of or relating to the essential nature of a thing: inherent

specifies the intrinsic properties of a job.

~~Functional parameter specifies the~~ Resource parameter gives us its resource requirement.

Interconnection parameter describes how it depends on other jobs & how other jobs depends on it.

Fixed, Jittered and sporadic Release time:-

Release Time of a job is represented by r_i of each job J_i.

r_i^- is called earliest release time.

r_i^+ is called latest release time.

If there is range like $[r_i^- \; r_i^+]$, it is called jitter response time.

for example:-

The earliest release time $r_i^- = 10$ sec

The latest release time $r_i^+ = 15$ sec

If the given release time falls in between this range [10, 15], it is jiter.

Fixed Release time is the predefined fixed time of release.

If for all practical purposes, we can approximate the actual release time of each job by its earliest or latest release time, then we say that the job has a fixed release time.

Sporadic jobs or aperiodic jobs :-

Almost every real-time system is required to respond to external events which occur at random instant of time. When such an event occurs, the system executes a set of jobs in response. The release times of these jobs are not known until the event triggering them occurs. These jobs are called sporadic jobs or aperiodic jobs because they are released at random time instants.

For example :- the pilot may disengage the autopilot system at any time. When this occurs the autopilot system changes from

cruise mode to standby mode. The jobs that execute to accomplish this mode change are sporadic jobs.

- The release times of sporadic and aperiodic jobs are random variables. [The model of system gives the probability distribution $A(x)$ of the release time of such job, or when there is a stream of similar sporadic or aperiodic jobs, the probability distribution of interrelease time (i.e. the ~~release~~ length of the time interval between the release times of two consecutive jobs in the stream)]. [$A(x)$ gives us the probability that the release time of the job is at or earlier than x (or the interrelease time of the stream of jobs is equal to or less than x) for all valid values of x. Rather than speaking of release times of ~~(sporadic)~~ aperiodic jobs, we sometimes use the term arrival times (or interarrival time) which is commonly used in queueing Theory. An aperiodic job arrives when it is released. $A(x)$ is the arrival time distribution (or interarrival time distribution).]

Execution Time

Another temporal parameter of a job, J_i, is its execution time, e_i.

e_i is the amount of time required to complete the execution of J_i when it executes alone and has all the resources it requires.

We say that execution time e_i of the job J_i is in the range $[e_i^-, e_i^+]$ where e_i^- & e_i^+ are the minimum & maximum execution time of J_i, respectively.

The term execution time e_i of each job J_i specifically means its maximum execution time.

Note:- e_i^- and e_i^+ of every hard real-time job J_i is known. (assumption)
But actual execution time is unknown

→ Hard-real time systems are safety-critical and are designed and implemented in such a way that the variations in job execution times are kept as small as possible.

→ Deterministic (approach) model is used & designed for execution time estimation.

→ Design the hard real-time system based on worst case processor time and resource requirements ~~even~~ and use the methods

and tools supported by the deterministic model to ensure that the hard timing constraints will surely be met at all times.

Periodic Task model :-

→ The periodic task model is a deterministic workload model.

→ It occurs in a regular or semi regular interval of time.

→ The example of periodic Task model is continuous image capturing in airport signaling. (Real time monitoring, digital control, constant bit-rate voice / video transmission.)

→ Many scheduling algorithm based on this model.

Periods, Execution Times and phases of Periodic Tasks

- **Periods :-**

The period P_i of a periodic task T_i is the minimum length of all time intervals between release time of consecutive job in T_i.

Execution Time :- The execution time of periodic task model is the maximum execution of all the jobs in it.

- **Phases :-**

The release time $r_{i,1}$ of the first job $J_{i,1}$ in each task T_i is called phases of T_i.

For convenience, we can use ϕ_i to denote phases of T_i. i.e. $\phi_i = r_{i,1}$.

Different tasks may have different phases. Some tasks are in phase, i.e. having the same phase.

Hyperperiod :-

H represents the least common multiple of periods P_i for $i = 1, 2, \ldots, n$. A time interval of length H is called a hyperperiod of the periodic tasks.

The maximum number N of jobs in each hyperperiod is equal to $\sum_{i=1}^{n} H/P_i$.

for eg:-

The length of hyperperiod of three periodic tasks with periods 3, 4 and 10 is 60.

Then, total no. of jobs in the hyper period is N,

So, $N = \sum_{i=1}^{3} \frac{60}{P_i}$

$= \frac{60}{3}^{20} + \frac{60}{4}^{15} + \frac{60}{10} = 41$ jobs

Utilization of Tasks :-

Utilization of Task T_i, is denoted by u_i and $u_i = e_i/p_i$

u_i is equal to the fraction of time with truely periodic task with period P_i and execution time e_i.

It is the upper bound to the utilization of

Run cmd in admin → Simgr␣ − rearm
Install window activator

~~task~~ any task modeled by T_i.
The total utilization U of all the tasks in the system is the sum of utilizations of the individual tasks in it.

for example:-

if $e_1 = 1$
$e_2 = 1$
$e_3 = 3$

and their periods are $p_1 = 3$, $p_2 = 4$
and $p_3 = 10$

Then Their utilizations are

$u_1 = 0.33$
$u_2 = 0.25$
$u_3 = 0.3$

The total utilization of the task is 0.88; this idicates that these tasks can keep a processor busy at most 88% of the time.

Note:- Deadline of a job, $D_i \leq P_i$

Aperiodic & Sporadic Tasks:-

In periodic task model, the workload generated in response to the unexpected events (ie. aperiodic jobs or sporadic jobs) is captured by aperiodic and sporadic tasks. Each aperiodic and

aperiodic or
sporadic task is a stream of sporadic jobs, respectively. The interarrival times between consecutive jobs in such a task may vary widely and, in particular, can be arbitrarily small. [The jobs in each task are model the work done by the system in response to events of the same type.] for eg:- The jobs that execute to change the detection threshold of the radar system are in one task; the jobs that changes the operation mode of the autopilot are in one task; [and the jobs that process sporadic data messages are in one task, and so on.]

→ Within any hyperperiod of the periodic tasks no periodic tasks are added or deleted.

→ We say that a task is aperiodic if the jobs in it have either soft deadlines or no deadlines. Example, the task to change radar's sensitivity.

→ Tasks containing jobs that are released at random time instants and have hard deadlines are sporadic tasks. Sporadic tasks are treated as hard real-time tasks. [Our primary concern is to ensure that these

deadlines are always met; minimizing their response times is of secondary importance.)

PRECEDENCE CONSTRAINTS AND DATA DEPENDENCY:-

Data and control dependencies among jobs may constraint the order in which they can execute. In classical scheduling theory, the jobs are said to have precedence constraints if they are constrained to execute in some order. otherwise, if the jobs can execute in any order, they are said to be independent.

for eg:- In a radar surveillance system, the signal-processing task is the producer of track records, while the tracker task is the consumer. In particular, each tracker job processes the track records generated by a signal-processing job. The designer may choose to synchronize the tasks so that the execution of the k^{th} tracker job does not begin until the k^{th} signal-processing job completes. The tracker job is precedence constrained. In general, a consumer job has precedence constraints.

Precedence Graphs and Task Graphs

$<$ ← partial order relation called a precedence relation.

→ A job J_i is a ~~precedence~~ predecessor of another job J_K (and J_K a successor of J_i) if J_K cannot begin execution until the execution of J_i completes.

i.e. $J_i < J_K$ (in notation)

J_i is an immediate predecessor of J_K if there is no other J_i such that $J_i < J_i < J_K$.

J_K ← immediate successor of J_i

→ A job with predecessors is ready for execution when the time is at or after its release time and all of its predecessors are completed.

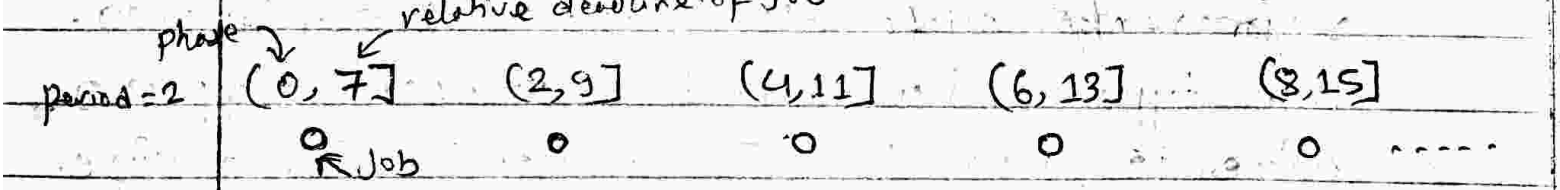
* A classical way to represent the precedence constraints among jobs in a set J is by a directed graph $G = (J, <)$. Each vertex in this graph represents a job J_i in J. There is a directed edge from the vertex J_i to the vertex J_K when the job J_i is an

immediate predecessor of the job J_k. This graph is called precedence graph.

→ A graph $G = (J, <)$ is called a precedence graph if there is a directed edge from the vertex J_i to the vertex J_k when the job J_i is an immediate predecessor of the job J_k.

(J: a set of jobs.)

→ A task graph, which gives us a general way to describe the application system, is an extended precedence graph.
As in precedence graph, the vertices in a task graph represents jobs. They are circles or squares.



In this figure, the task has a phase 0, period 2 and relative deadline 7. The jobs, here, are independent, so there are no any directed edges. Thus, the jobs released in later periods are ready for execution as soon as they are released even though some jobs released earlier is not yet complete.

figure :- Precedence graph.

The vertices represent jobs in a periodic task with phase 2, period 3, and relative deadline 3. The jobs in it are dependent, the 1st job is the immediate predecessor of the second job, and the second job is the immediate predecessor of the third job and so on.

[Many kinds of interactions and communications among jobs are not captured by a precedence graph but can be captured by a task graph. Unlike a precedence graph, a task graph may contain different types of edges that represent different types of dependencies. The type(s) dependency represented by an edge is given by the type(s) of the edge. The types of an edge connecting two vertices and other parameters of the edge are interconnection parameters of the jobs represented by the vertices.]

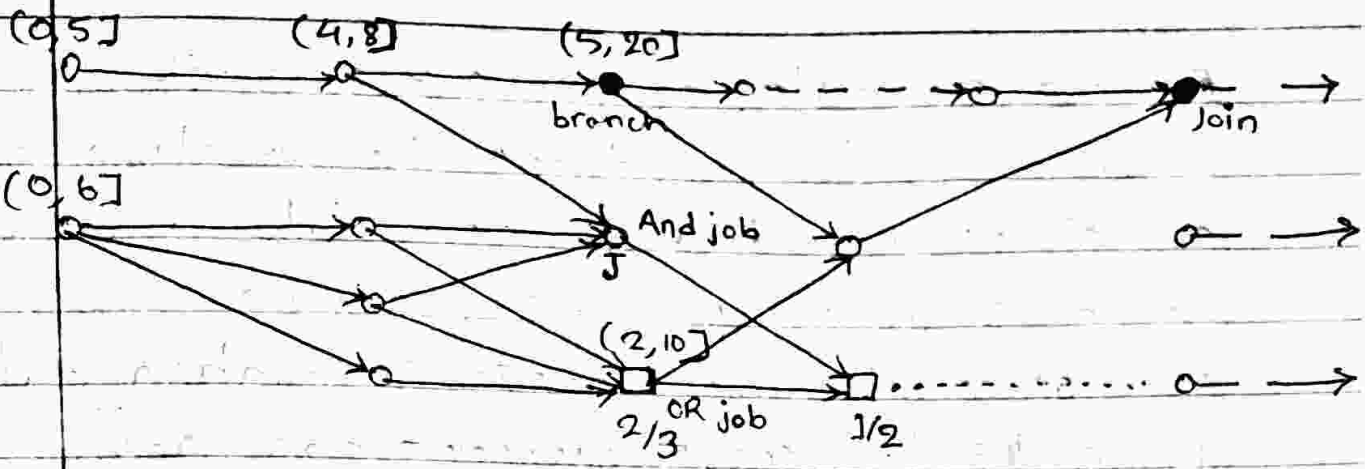


figure 3-1: Example of task graphs.

Q.No.3
Q.P 68

Data Dependency:-

In a task graph, data dependencies among jobs are represented explicitly by data dependency edges among jobs. There is a data dependency edge from a vertex J_i to vertex J_k in the task graph if the job J_k consumes data generated by J_i or the job J_i sends message to J_k.

A parameter of an edge from J_i to J_k is the volume of data from J_i to J_k.

[Note:- Data dependency can not be captured by a precedence graph. In ~~real-time~~ systems, jobs communicate via shared data. Often time, the designer chooses not to synchronize producer and consumer jobs. Rather, each producer places the data generated by it in a shared address space to be used by the consumer at any time. In this case

As an example, in an avionics system, the navigation job updates the location of the airplane periodically. These data are placed in a shared space. Whenever the flight management job needs navigation data, it reads the most current data produced by the navigation job. There is a precedence constraint between the navigation job and flight management job.

Other types of dependencies

→ Temporal Dependency
→ AND/OR precedence constraints
→ Conditional Branches
→ Pipeline Relationship.

a) Temporal dependency :-

→ Some jobs may be constrained to complete within a certain amount of time relative to one another. We call the difference in the completion times of two jobs the temporal distance between them.

→ Jobs are said to have a temporal distance constraints if their temporal distance must be no more than some finite value.

→ such jobs may or may not have deadlines.

→ For Example :- Person and audio should

match, when person wants to have lip synchronization. for this the value of temporal distance should not exceed 160 msec.

→ In the task graph, temporal distance constraints among jobs are represented by temporal dependency edges. There is a temporal dependency edge from a vertex J_i to a vertex J_k if the job J_k must be completed within a certain time after J_i completes.

b) AND/OR Precedence constraints :-

AND

In the classical model, a job with more than one immediate predecessor must wait until all its immediate predecessors have been completed before its execution can begin. Then such a job is called AND job and with dependency of AND precedence constraints.

→ In the figure 3-1, the job labeled J is an example of AND job. All three of its immediate predecessors must be completed before J can begin execution. And jobs are represented by unfilled circles in the task graph.

OR job

An OR job is one which can begin execution at or after its release time provided one or some of its immediate predecessors has been completed.

In figure 3-1 we represent OR jobs by square vertices. The one labeled 2/3 can begin execution as soon as two out of its three immediate predecessors completed.

→ In-type of a job tells us that all its immediate predecessors must complete before its execution.

Note:- AND job (By default)

And in-type jobs are ↑

© Conditional Branch :-

→ Conditional Branch can be modeled by a task graph that has edges expressing OR constraints. Only one of all the immediate successors of a job whose out-going edges express OR constraints is to be executed. Such a job is called a branch job. In a meaningful task graph, there is a join job associated with each branch job. In figure 3-1, these jobs are represented by filled circles.

→ The subgraph that begins from a vertex representing a branch job and ends at the vertex representing the associated join job is called a conditional block. Only one

conditional branch in each conditional block is to be executed.

→ The job parameter out-type tells whether all the job's immediate successors are to be executed.

→ The default value of out-type parameter of every job is AND. [That is all its immediate successors are to be executed.] But the out type of a branch job is OR, because only one of its immediate successors must be executed.

ⓓ Pipeline Relationship :-

→ A dependency between a pair of producer-consumer jobs that are piped can theoretically be represented by a precedence graph. In this graph, the vertices are the granules of the producer or consumer. Each ~~granule~~ granule of the consumer can begin execution when the previous granule of this job and the corresponding granule of the producer job have completed.

In the task graph, we represent a pipeline relationship between jobs by a pipeline edge, as exemplified by the

dotted edge between the jobs in the right bottom corner of the graph in figure 3-1. There is an edge from J_i to J_k if the output of J_i is piped into piped into J_k and the execution of J_k can proceed as long as there are data for it to process.

Functional Parameters :- Main functional parameter include :-

a) premptivity of jobs
b) Criticality of jobs
c) Optional Executions
d) Laxity Type and Laxity function.

a) Primitivity of jobs,

The scheduler may suspend the execution of a ~~job~~ less urgent job and give the processor to a more urgent job. Later, when the more urgent job completes, the scheduler returns the processor to less urgent job so the job can resume execution. This interruption of job execution is called preemption.

nevigation → The process of monitoring and controlling the movement of a craft or vehicle.

→ A job is preemtable if its execution can be suspended at any time to allow the execution of other job and, later on, can be resumed from the point of suspension. computation job that execute on CPUs are ~~called~~ example of preemptable job.

→ A job is non preemtable if it must be executed from start to completion without interruption.

b) Criticality of jobs:-

In any system, jobs are not equally important. The importance (or criticality) of a job is a positive number that indicates how critical the job is with respect to other job; the more critical the job is, the larger its importance.
For example, in a flight control & management system, the job that controls the flight of the aircrayt is more critical than the ~~neg~~ nevigation job that determine the critical position. The cabin air flow and temperature controls jobs are more critical

than the job that run in flight movies. In the model of this system, the designer may give these jobs different importance values. In this way, the different degree of criticality of the jobs are explicitly specified.

c) Optional Executions:-

(Execute job optionally)
It is possible to structure an application so that some jobs or portions of jobs are optional. If an optional portion of job complete late or is not executed at all, the system performance may degrade but functions satisfactorily.

For example:-
Collision avoidance system. The collision avoidance system may still function satisfactorily if it skips warning module in some extent.

d) Laxity Type and Laxity function:-
indicates whether its timing constraints are hard or soft.

The laxity type of a job indicates whether its timing constraint are soft or hard.
Laxity type of a job is sometimes supplemented by a usefulness function.
This function gives the usefulness of the result produced by the job as a function of its tardiness.

Resource Parameters :-

a) preemtivity of resources
b) Resource graph.

a Preemptivity of Resources :- A resource parameter is primitive in nature.

→ In other words, once a unit of a non-preemptable resource is allocated to a job, other job needing to the unit must wait until the job completes its use.

→ A resource parameter is non-preemptive if each unit of resource is constrained to be used serially.

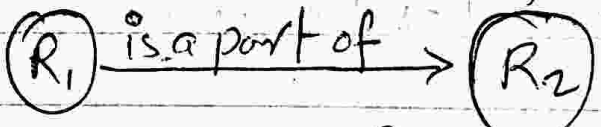
→ otherwise, if job can use every unit of a resource in a interleaved fashion the resource is preemtable.

(5) Resource graph :-

- A resource graph can describe the configuration of the resources.
- In a resource graph, there is a vertex R_i for for every processor or resource R_i in the system.
- The attribute of the vertex are the parameters of the resource.
- The resource type of a resource tells us whether the resource is a processor or a passive resource.

There are two types of edges in the resource graph which represents the relationship among resources.

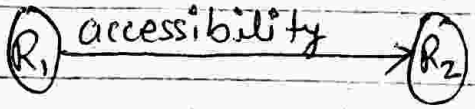
An edge from vertex R_i to another vertex R_K mean that R_K is component of R_i. This edge is an "is_a_part_of" edge.



→ Some edges in Resource graph represent connectivity between components.

These edges are called accessibility edges. If there is a connection between two CPUs in two computers, [we can draw accessibility edge.] then each CPU is

accessible from the other computer, and there is an accessibility edge from each computer to the CPU on the other computer.



Q. No. 10
2068

Scheduling Hierarchy

The application system is represented by a task graph as in figure. It gives the processor time and resource requirements of jobs, the timing constraints of each job, and the dependencies of jobs.

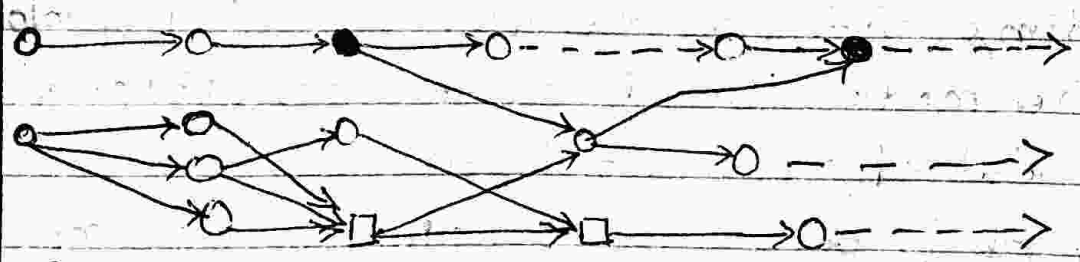


fig:- Task graph

The resource graph describes the amount of the resource available to execute the application system, the attributes of the resources, and the rules governing their usage.
Between them are the scheduling and resource access control algorithms used by the operating system.

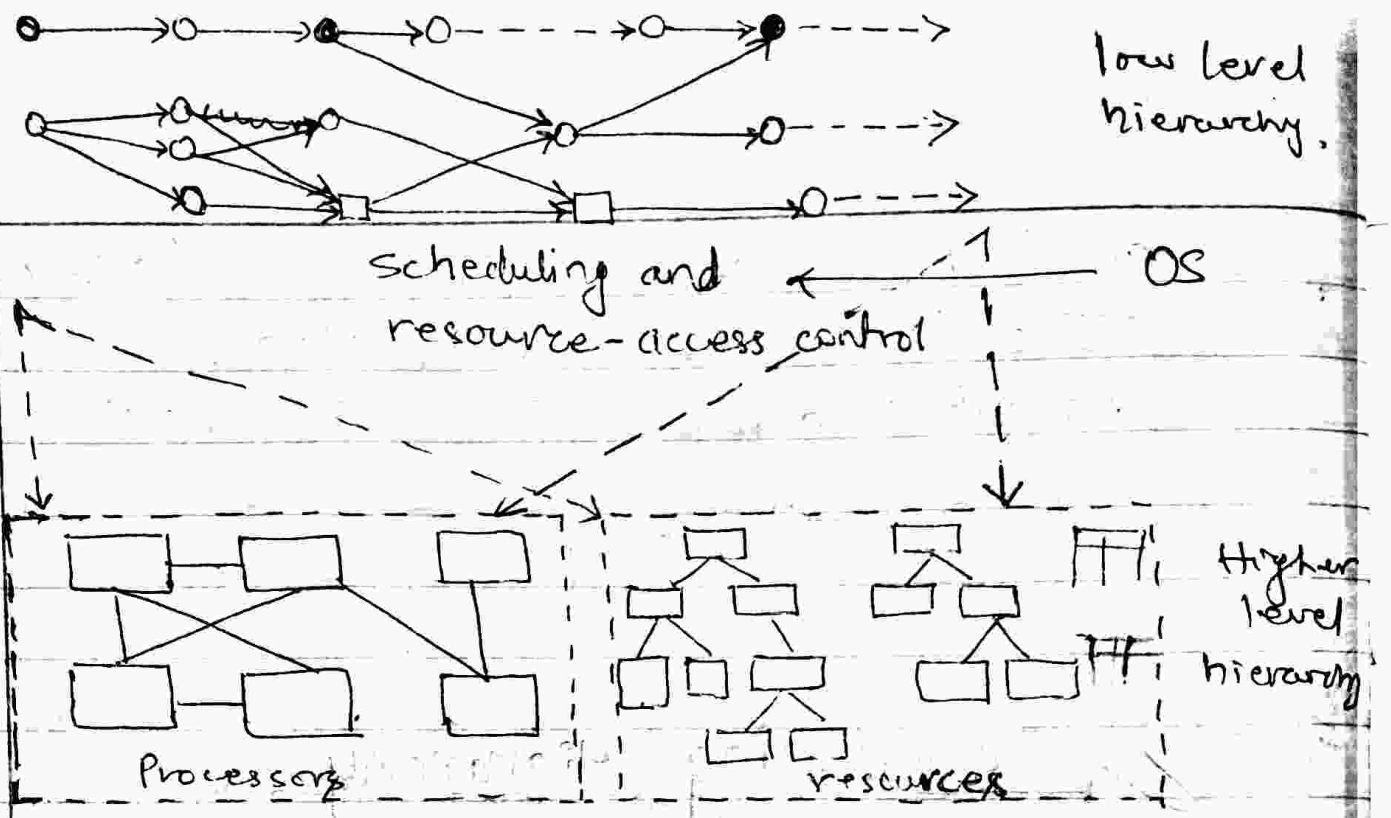


figure :- Model of Real-Time systems.

a) Scheduler and Schedule

Jobs are scheduled and allocated resources according to a chosen set of scheduling algorithms and resource access-control protocols. The module which implements these algorithms is called the scheduler.

A job is said to be scheduled in a time interval on a processor if the processor is assigned to the job, and hence the job executes on the processor in the interval.

The total amount of processor time assigned to a job ~~execute~~ according to a schedule is the total length of all ~~the scheduler~~ the time intervals during which the job scheduled on some processor.

A schedule means an assignment of all the jobs in the system on the available processors produced by scheduler.

We assume that a scheduler works correctly and always produces only valid schedules.

A valid schedule satisfies the following conditions:-

1. Every processor is assigned to at most one job at any time.
2. Every job is assigned to at most one processor at any time.
3. No job is scheduled before its release time.
4. Depending on the scheduling algorithm used, the total amount of processor time assigned to every job is equal to its maximum or actual execution time.
5. All the precedence and resource usage constraints are satisfied.

Note:- An implicit assumption is that jobs do not run in parallel on more than one processor to speed up their execution.

b) Feasibility, Optimality, and Performance measures:

A valid schedule is a feasible schedule if every

job completes by its deadline (or, in general meets its timing constraints). A job is schedulable according to a scheduling algorithm if when using the algorithm, the scheduler always produces a feasible schedule.

Optimality ⇒ A Hard real time scheduling algorithm is optimal if the algorithm (scheduler) always produces a feasible schedule if the given set of jobs has feasible schedules.
Conversely, if an optimal algorithm fails to find a feasible schedule, we can conclude that the given set of jobs can not ~~be~~ feasibly be scheduled by any algorithm.

Performance measures ⇒ Other commonly used performance measures include the maximum and average tardiness, lateness, and response time and miss, loss and invalid rates.

1. Lateness ⇒ The lateness of a job is the difference between its completion time and its deadlines. Unlike the tardiness of a job which never has negative values, the lateness of a job which completes early is negative, while the lateness of a job which completes late is

positive. Sometimes we want to keep jitters in the completion times (small) by using algorithms that (try to) minimize the average absolute lateness of jobs. for eg:- Transmission of packets in a packet-switched network.

makespan

2. for jobs having same release time and deadline a scheduling algorithm that produce a schedule with shorter makespan is better. A makespan is the response time of a job (which completes last among all jobs) that is equal to the response time of the set of jobs as a whole. So, if the makespan is less than or equal to the length of their feasible interval, the job can meet their deadline.

average Response time

3. The most frequently used performance measures for jobs that have soft deadlines is their average response time. The smaller the average response time, the better the algorithm. (just as for general-purpose, interactive systems.) In a system that have hard and soft real time jobs, the objective of scheduling is to minimize the average response time of soft real time jobs to ensure deadlines and to delay the execution of hard real time jobs to improve the response time of jobs with soft deadlines.

miss rate, loss rate, invalid rate.

4. In soft-Real time applications, it is acceptable to complete some jobs late or discard late jobs. So the suitable performance measures include the miss rate ~~and~~, loss rate. and invalid rate.

* **Miss rate :-**
The percentage of jobs that are executed but completed too late.

* **loss rate :-**
The percentage of jobs that are ~~too late~~ discarded, that is, not executed at all.

→ When a scheduler is impossible to complete all jobs, it discards some jobs. This increases the loss rates but completes more jobs in time. Thus it reduces the miss rate.

• Similarly reducing the loss rate may lead to miss rate increase.

→ So, miss rates and loss rates are so adjusted that they donot cross some threshold values. This trade-off is captured by invalid rate.

* **Invalid rate :-** is the sum of miss rate and loss rate to give the percentage of all jobs that do not produce a useful results. We want to minimize the invalid rate.

for example:- If the jobs are transmission of real time packets, the miss rate gives the percentage of packets arriving at the receiver too late, the loss rate gives the the percentage of packets discarded en route, and the invalid rate gives the percentage of packets that are not useful to the application.

Interaction among Schedulers :-

Scheduling the application system on the underlying processors and resources, has a hierarchy of schedulers. This scheduling hierarchy arises for two reasons :-

1. Some processors and resource used by the application system are logical resources (not physical), which must be scheduled on physical resources. [The algorithm used for this purpose are typically different from the ones used to schedule the application system.] A scheduler that schedules a logical resource may be different from the scheduler that schedules the application system using the resource.

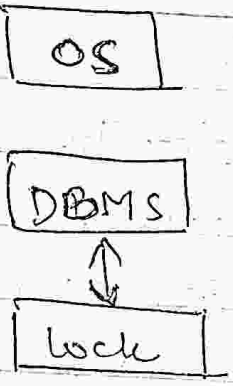
2. A job may model a server that executes on behalf of its client jobs. The time and resources allocated to the server job must in turn be allocated to its client jobs. Again, the algorithm used by the server to schedule its clients may be different from the algorithm used by the operating system to schedule the server with other servers.

Example

Database locks (As resource) are implemented by DBMS whose execution must be scheduled on one or more processors. The scheduler that schedules the ~~database~~ DBMS may be different from the scheduler that schedules the application system using the locks. (The schedulers most likely to use different algorithms). Now we have two level of scheduling :-

→ In the higher level, the application system is scheduled on the resources.

→ In the lower level, the jobs that execute in order to implement the resource are scheduled on the processors and resources (needed by them.)



High level scheduler

Scheduler (low level scheduler)

Surprize test-1

1. What do you mean by processor and resources? (5)
2. How do you define preemptivity of Resource? (5)
3. Define AND/OR constraints of a job. (5)
4. Define Precedence graph or Task graph in RTS. (5)

Unit Test-2
2070-9-16

Time 1:20 hrs P.M. 15 F.M. 25.

6. How do you mean by valid schedule and what
3. What are the conditions for a valid schedule? (1+5)
4. What do you mean by Aperiodic and Sporadic tasks? How they differ? (4+1)
5. Write short notes on: (5×1 = 5)
   a) Phase of a periodic task.
   b) Execution time of a " "
   c) Period of a periodic task.
   d) Hyper period of Periods.
   e) Invalid rate.

Full Marks 80
Pass Marks 32
Time 3 hr.

1st mid term Exam - 2070
(Sixth Sem)

A) Long question (2x10 = 20)

(1) Define and describe digital control with well labeled diagram.

(2) Describe Radar signal processing and tracking system with a neat labeled diagram.

(B) Answer any Ten. (6x10 = 60)

(1) Why do we need multirate system in Real time system?

(2) How does nearest neighbouring algorithm role in Data aquisition tracking?

(3) Differentiate between absolute and relative temporal consistency in Real time database?

(4) Differentiate between Hard and soft real time systems. Given example of each.

(5) Consider a system used to monitor and control several heating furnaces that takes 10 ms to initialize when turn on.

(6) Define AND/OR constraint of a job.

(7) What is meant by a valid schedule and what are the conditions for

a valid schedule? (1+4)

(7) How many jobs a hyperperiod (H) comprises if the periods are 4, 6 and 9? If the execution times are 2, 2 and 5 respectively then what are the utilizations of the individual tasks? (3+3)

(8) Define Data dependency and pipeline relationship. (3+3)

(9) What do you mean by processor and resource in RTS?

(10) Write short any Three notes: (3×2=6)

a) Laxity type
b) Preemptivity of a job.
c) tardiness of a job.
d) Phase of a periodic task.