REAL TIME SYSTEMS
(TCS-041)

UNIT I INTRODUCTION

* Real time systems : is required to complete its work and deliver its services on timely basis.
Example of real time systems: includes digital control, command and control, signal processing and telecommunication systems.

* Digital Control :
Many real time systems are embedded in sensors and actuators, and functions as digital controllers.
Fig shows a system. The term plant in the block diagram refers to a controlled system, for example, a brake, an aircraft, a patient. The state of the plant is monitored by sensors and can be changed by actuators.

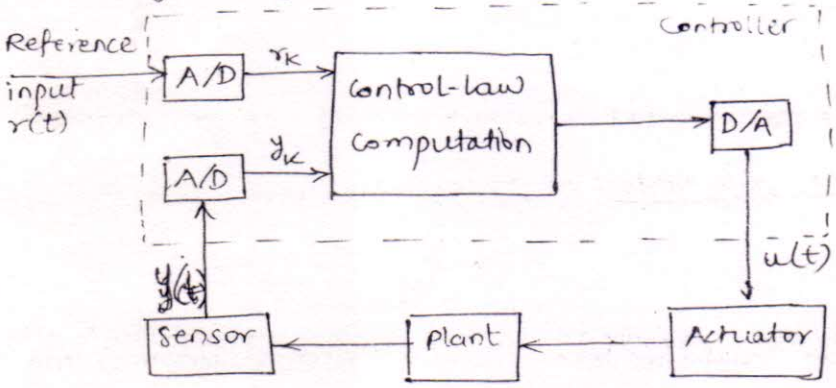

Fig. A digital Controller

Sampled data system: The analog version is then transformed into a digital version (i.e. discrete-time and discrete-state). The resultant controller is a sampled data system.

For example, : we consider an analog single-input/single-output PID (Proportional, Integral, and Derivative) controller.
The analog sensor reading $y(t)$ gives the measured state of the plant at time t. Let $e(t) = r(t) - y(t)$ denote the difference between the desired state $r(t)$ and the measured state $y(t)$ at time t.
The output $u(t)$ of the controller consists of three terms:
a term that is proportional to $e(t)$, a term that is proportional to the integral of $e(t)$ and a term that is proportional to the derivative of $e(t)$.
Sampled values y_k and r_k for $k = 0, 1, 2, \ldots$ which A/D converters produces by sampling and digitizing $y(t)$ and $r(t)$ periodically every T units of time. $e_k = r_k - y_k$ is the k^{th} sample value of $e(t)$.

we can approximate the derivative of $e(t)$ for $(k-1)T \leq t < kT$ by $(e_k - e_{k-1})/T$ and use the trapezoidal rule of numerical integration to transform a continuous integral into a discrete form. The result is the following incremental expression of the kth output u_k:

$$u_k = u_{k-2} + \alpha e_k + \beta e_{k-1} + \gamma e_{k-2} \longrightarrow 1.1$$

α, β and γ are proportional constants; they are chosen at design time.

from eq^n (1.1), we can see that during any sampling period (say the kth) 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.

<u>selection of sampling period:</u>

The length T of time between any two consecutive instants at which $y(t)$ and $r(t)$ are sampled is called the <u>sampling</u> period. T is key design choice.

<u>Multirate system</u>: Multirate system is monitored by multiple sensors and controlled by multiple actuators. Because different state variables may have different dynamics, the sampling periods required to achieve smooth responses from the perspective of different state variables may be different.

High-Level Controls

Controllers in a complex monitor and control system are typically organized hierarchically.

Example: a patient care system may consist of microprocessor-based controllers that monitor and control the patient's blood pressure, respiration, glucose, and so forth.

Example of control Hierarchy

* Issues in Real-Time computing
  - Must be much more reliable.
  - proper task scheduling,
  - much more specific in their applicati.
  - The consequences of their failure are more drastic.

Architectural issues:

1) Processor architectural issue,
2) Network architectural issue,
3) Architectures for clock synchronization,
4) Fault tolerance and reliability evaluation.

Operating System Issues:

1) Task assignment and scheduling,
2) Communications protocols,
3) Failure management and recovery,
4) clock synchronization algorithms,

Other issues

* programming Languages, (greater control over timing & need to interface special purpose device)
* Databases, (stock market, air-line reservations, artificial intelligence)
* Performance measures.

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 ~~exert~~ exerting on 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.

The output produced by the controller during the kth sampling period is given by.

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

The constants α and β_i's are chosen at design time. x_i is the value of the state variable in the ith sampling period. During each sampling period, the controller must compute an estimate of x_k from measured values y_i for $i \le k$.

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.

For example.

A simple monitor system, which takes a measured value y_k every sampling period k in order to estimate the value x_k of a state variable. Suppose that starting from time 0, the value of this state variable is equal to a constant x. Due to noise the measured value of y_k is equal to $x + \epsilon_k$, where ϵ_k is a random variable whose average value is 0 and standard deviation is σ_k. The Kalman filter starts with the initial estimate $\tilde{x}_1 = y_1$ and computes a new estimate each sampling period.

Specifically, for $k>1$, the filter computes the estimate $\tilde{x}_k$ as follows:

$$\tilde{x}_k = \tilde{x}_{k-1} + K_k(y_k - \tilde{x}_{k-1}) \longrightarrow ①$$

where

$$K_k = \frac{P_k}{\sigma_k^2 + P_k} \longrightarrow ②$$

is called Kalman Gain.

and P_K is the variance of the estimation error $\tilde{x}_k - x$; the latter is given by $P_K = E[(\tilde{x}_K - x)^2] = (1-K_{K-1})P_{K-1}$ → 3

ATC (Air Traffic Control),

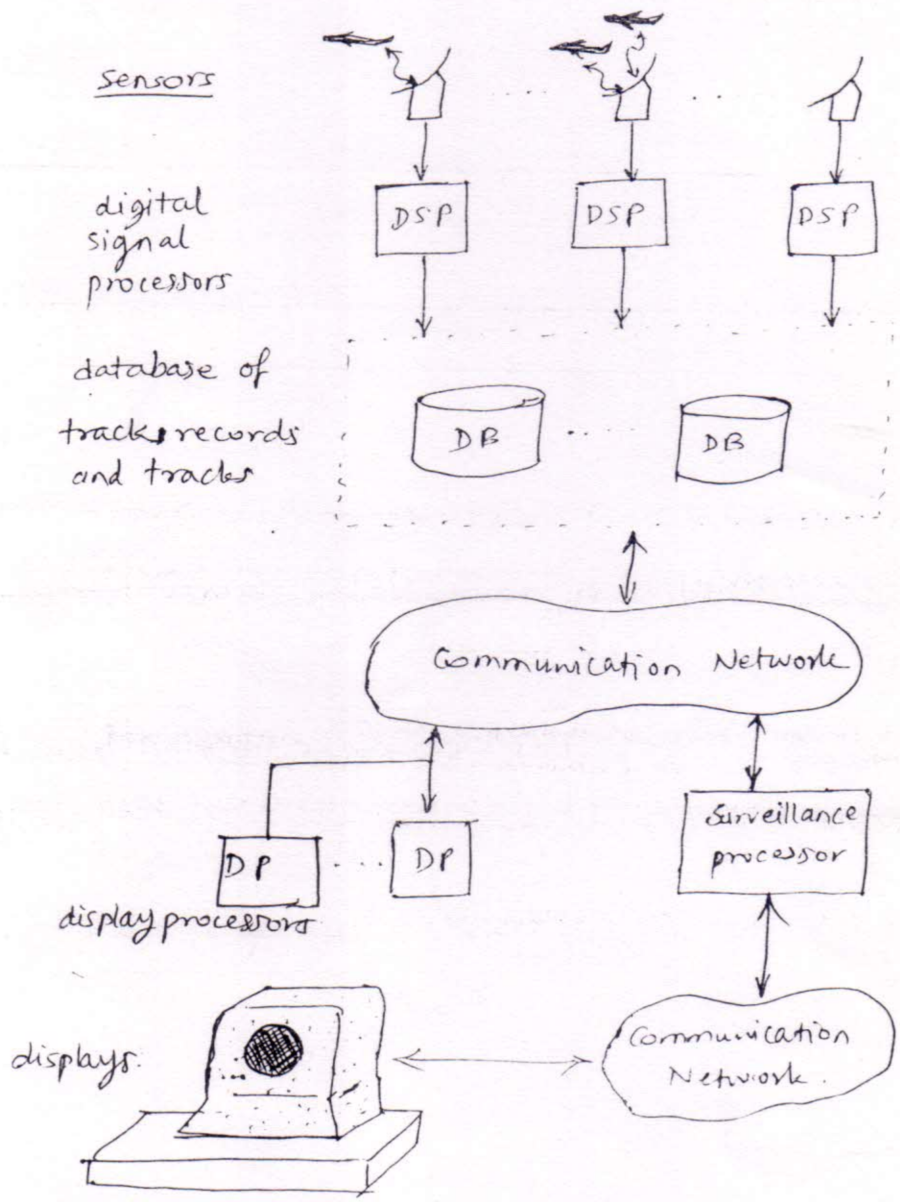

Fig: An architecture of air traffic control system.

- Guidance and Control
- Complexity and Timing Requirements,
- Other Capability: partial control of devices, operator interactions.
- Real time Command and Control.
- Tracking
- Gating. , -complexity and timing requirements.

Real-time databases:

Real time database contains data objects called image-objects, that represents real-world objects. The attributes of an image object are those of the represented real-world object.

For example: An air traffic control database contains image objects that represents aircraft in the coverage area. The attributes of such an image object include the position and heading of the aircraft. The values of these attributes are updated periodically based on the measured values of the actual position and heading provided by the radar system.

* Absolute temporal consistancy,
* Relative temporal consistancy,
* consistancy models.

Hard Vs Soft Real Time System

	Hard RTS	Soft RTS
1) Safety	often critical	non-critical
2) Peak load perfomance	predictable	degraded.
3) Data file size	small / medium	Large.
4) Data integrity	short term	long term
5) Redundancy time	Hard requised	soft requised.
6) Example	Air Traffic Control	Telecom (Voice)
7) Error detection	Autonomous	user-assisted.

Industrial applications of Real time system:

1) Metal Industry applications (Mechanical and Manufacturing Engineering).
2) water plants,
3) Aviation & space applications,
4) petrochemical applications
5) Data Communication applications (Electrical & Computer Engineering)

Real time versus Conventional Software:

1) Timing constraints,
2) Concurrency,
3) Reliability,
4) General purpose vs application specific,
5) Testing & certification.

* Hard versus Soft Real-Time Systems *

Jobs and Processors:

Each unit of work that is scheduled and executed by the system a job and a set of related jobs which jointly provide some system function a task.
Hence, the computation of a control law is a job, so is the computation of FFT of sensor data, the transmission of a data packet, or the retrieval of a file, and so on.

Every job is executed on some resource.

For example, the jobs mentioned above execute on a CPU, a network, and a disk, respectively. These resources are called servers in queuing theory literature and sometime, active resources in real-time systems literature.
The server is called as processors.

~~[illegible]~~, Release Times:→

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.

Deadline

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

Timing Constraints: A timing constraints of a job can be specified in terms of its release time and relative or absolute deadlines.

Hard and Soft Timing Constraints:

It is common to divide timing constraints into two types: hard and soft.

Common Def^{n}. 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.

Soft: The late completion of a job that has a soft deadline is undesirable. However, a few misses of soft deadlines do not serious harm; only the overall system performance becomes poorer and poorer when more and more jobs with soft deadlines complete late.

Processors and Resources

~~Processors and~~ System resources are divided into two types

1) Processors and

2) Resources.

* Processors: These are often called servers and active resources; computers, disks, transmission links and database servers are ~~all~~ examples of processors.

Two processors are same type if they are functionally identical. and can be used interchangeably.

P denotes processor(s). $P_1, P_2, P_3 \ldots P_m$. If m processors are there.

* Resources: It means passive resources

Example: memory, sequence numbers, database locks

Example: a computation job may share data with other computations, and data may be guarded by semaphores. we model each semaphores as resource.

Temporal parameters of Real-time Workload

We typically assume that many parameters of hard real-time jobs and tasks are known at all times; otherwise, it would not be possible to ensure that the system meet its hard-real time requirements. The number of tasks (or jobs) in the system is one such parameter.

Each job J_i is characterized by its temporal parameters, functional parameters, resource parameters, and interconnection parameters. Its temporal parameters tell us its timing constraints and behaviour release time, absolute and relative deadline of job J_i; these are temporal parameters.

Fixed, jittered and sporadic Release times

In some system we do not know the actual release time r_i of each job J_i; only that r_i is in a range $[r_i^-, r_i^+]$. r_i can be as early as the earliest release time r_i^- and as late as the latest release time r_i^+.

Some models assume that the only range is known and call this range the jitter in r_i, or release-time jitter.

Sporadic jobs or aperiodic jobs: are the jobs that are released at random time instants.

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.

Hence the value of this parameter depends mainly on the complexity of the job and speed of the processor and on how the job has been scheduled.

e_i^- & e_i^+ : minimum and maximum execution time.

Periodic task Model:

The periodic task model is a well-known deterministic workload model

Periods, Execution times and Phases of Periodic Tasks.

Periodic task is denoted by T_i is a sequence of jobs.
The Period P_i of the periodic task T_i is the minimum length of all time intervals between release times of consecutive jobs in T_i.

Execution time: is the maximum execution time of all the jobs in it.

Aperiodic ~~sporadic~~ Tasks:

A task is aperiodic if the jobs in it have either soft deadlines or no deadlines.
Example: The task to adjust Radar's sensitivity.

Sporadic tasks:

Tasks containing jobs that are released at random time instants and have hard deadlines are sporadic tasks.

Precedence constraints and data dependency

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.

Precedence Graph

A partial-order relation $<$, called a precedence relation.

A job J_i is a 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. A short hand notation ~~for~~ to state this fact is $J_i < J_K$.

J_i is the immediate predecessor ~~and~~ of J_K (and J_k is the immediate successor of J_i)

If $J_i < J_K$ and there is no other job J_j such that $J_i < J_J < J_K$.

Two jobs J_i & J_K are independent when neither $J_i < J_K$ nor $J_K < J_i$.

A classical way to represent ~~that~~ the precedence constraints among jobs in a set J is by a directed Graph $G = (J, <)$

Each vertex in this Graph represents a job in J.

Each vertex is called by the name of the job represented by it.

There is a directed ~~eg~~ edge from the vertex J_i to vertex J_K when the job J_i is an immediate predecessor of the job J_K.

This graph is called a precedence graph.

Example of precedence graph:

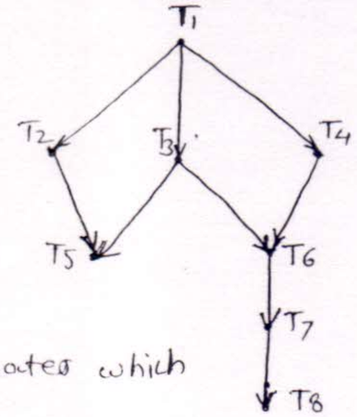

Fig: precedence graph

The arrow indicates which task has precedence over other task. The precedence task set of task T by $<(T)$; i.e. $<(T)$ indicates which tasks must be completed before T can begin.

e.g. We have

$$<(1) = \phi \qquad <(5) = \{1, 2, 3\}$$
$$<(2) = \{1\} \qquad <(6) = \{1, 3, 4\}$$
$$<(3) = \{1\} \qquad <(7) = \{1, 3, 4, 6\}$$
$$<(4) = \{1\} \qquad <(8) = \{1, 3, 4, 6, 7\}$$

we can also write $i < j$ to indicate that task T_i must precede task T_j. The precedence operator is transitive.

$$i < j \text{ and } j < k \Longrightarrow i < k$$

In some cases, > and < are used to denote which task has higher priority i.e. $i > j$ can mean that T_i has ~~p~~ higher priority than T_j

Data Dependency

Data dependency cannot be captured by a precedence graph. In many RTS, jobs communicate via shared data.

In an avionics system, the navigation job updates the location of the air-plane periodically. These data are placed in a shared space. whenever a flight management system needs ~~m may~~ navigation data, it reads the most current data produced by the navigation job. There is no precedence constraint between the navigation job and the flight management job.

In a task graph, data dependencies among jobs are represented by data dependency edges among jobs.

There is data-dependency edge from vertex J_i to vertex J_K in the task graph if the job J_K ~~continues~~ consumes data generated by J_i or the job J_i sends messages 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.