A Reference Model of Real Time Systems

Pei-Hsuan Tsai

CPS LAB

A reference model of real-time systems

- A good model abstracts the irrelevant details.
  - Focus on timing properties and resource requirement of system components
  - The operating system allocates the available system resources among them
- Each system is characterized by three elements:
  - A workload model that describes the applications supported by the system.
  - A resource model that describes the system resources available to the applications.
  - Algorithms that define how the application system uses the resources at all times.

CPS LAB

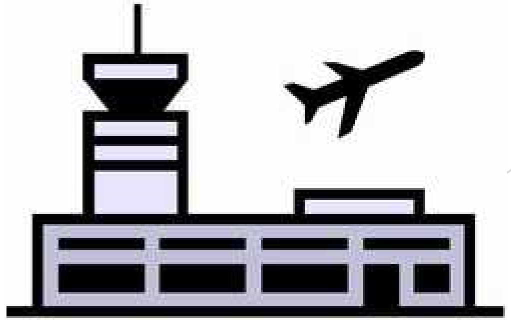
3.1 Processors and resources

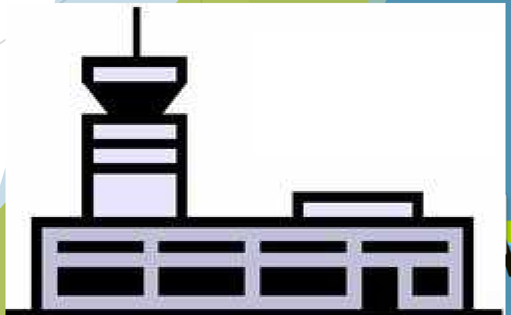
Resource Model

- We divide all the system resources into two major types: *processors* and *resources.*
- Processors are often called servers and active resources.
  - Ex: computers, transmission links, disks and database server.
- Resources are passive resources.
  - Ex: memory, sequence numbers, mutexes and database locks.

Processors

- Every job must have one or more processors in order to execute and make progress toward completion.
- *Types* of processors: two processors are of the same type if they are functionally identical and cane be interchangeably.
- Example:
  - Two transmission links with the same transmission rate between a pair of sender and receiver are the same type.
  - Processors in a Symmetrical Multiprocessor (SMP) system are of the same type.
- Processors that are functionally different cannot be used interchangeably are different types.
  - Ex: CPUs, transmission links and disks are of different types.
  - A transmission link connecting an on-board flight management system to the ground controller is a different type professor from the link connecting two air traffic control centers.
  - Even when the links have the same characteristics. They cannot be used interchangeable.
- The letter, P, denote processor(s).
- m processors in the system, $P_1, P_2, \ldots, P_m$.
- *Speed* of a processor.

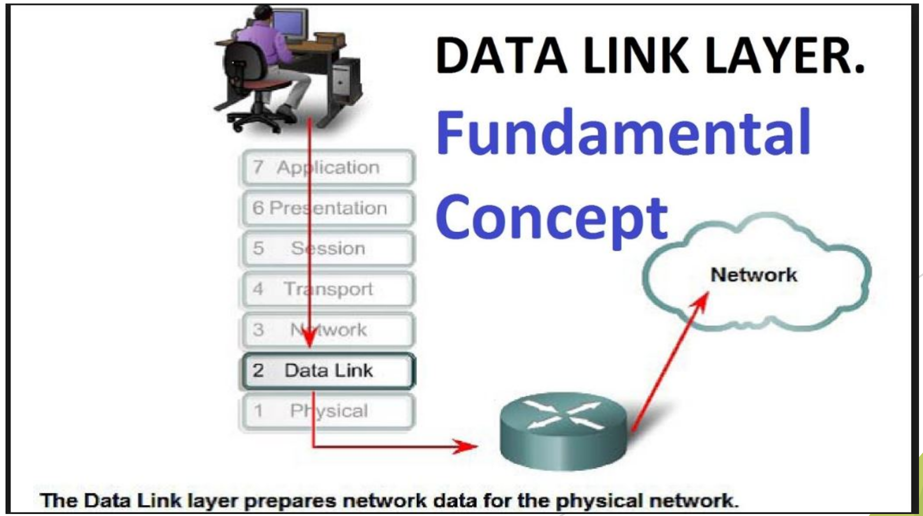




Resources

- Passive resources: memory, sequence number, mutexes and database locks.
- A job may need some resources in addition to the processor in order to the processor in order to make progress.
- A job takes to complete depends on the speed of the processor but does not depend on the speed of any resource it uses during execution.
- Examples:
  - A computation job may share data with other computations, and the data may be guarded by semaphores. We model each semaphore as a resource.
  - A data link that uses the sliding-window scheme to regulate message transmission. (Next page.)
-

REFERENCE: https://forum.huawei.com/enterprise/en/q-a-what-is-one-main-characteristic-of-the-data-link-layer/thread/455117-100181



CPS LAB

Resources

- Letter, *R*, to denote resources.
- Resources are reusable and not consumed during use.
- Each resource may have one or more units and used in mutually exclusive manner.
  - Ex: Sliding window of a data link has eight valid sequence numbers. →there are eight units of the sequence-number resource.
- A resource is *plentiful* if no job is ever prevented from execution by the lack of this resource. →need not be explicitly modeled.
- Memory is an essential type of resource. Omit it from our model whenever we can account for the speed of the memory by the speed of the processor.
  - Ex: The speed of the buffer memory in a packet switch by letting the speed of each input link equal the transmission rate of the link (the rate at which data can go in the buffer).

CPS LAB

No cookbook rules for modeling

- Sometimes model some elements of a system as processors and sometimes as resources, depending on how we will use the model.
  - Ex: I/O bus. Most of the time, I/O bus is modeled as a resource. When we want to study how long I/O activities may delay the completion of computation jobs that share the I/O bus, I/O bus may be modeled as a processor.
- A good model can give us better insight into the problem at hand.
- A bad model can clutter our mind with irrelevant details and man even give us a wrong point of view and lead to different scheduling and resource strategies.

3.2 Temporal parameters of real-time workload

- Parameters of hard real-time jobs and tasks are known at all the times; otherwise, it would not be possible to ensure that the system meet its hard real-time requirements.
  - Ex: the number of tasks (jobs) in the system is one such parameter.
- In many embedded systems, the number of tasks (jobs) is fixed as long as the system remains in an operation mode. It may change when the system operation mode changes.
  - Ex: Flight control system.
    - During cruise mode (飛行模式), the system has 12 tasks.
    - During landing, the number of tasks increase to 24 tasks.
- In some system, the number of tasks may change as tasks are added or deleted while the system executes.
  - Ex: Air traffic control system.
    - The number of tasks is added and deleted when the aircraft enter and leave the coverage area.
  - Nevertheless, the number of tasks with hard timing constraints is known at all the times.
  - The run-time system decide the admission and deletion of tasks requested by application system.

CPS LAB

Parameters of a job

- Each job J_i is characterized by 4 parameters:
  - Temporal parameters
  - Interconnection parameters
  - Functional parameters
  - Resource parameters
- Temporal parameters:
  - release time, r_i
  - absolute deadline, d_i
  - relative deadline, D_i
- *Feasible interval* of job$_i$: $(r_i, d_i]$; “(or)“→不包含，”[or]” →包含。

Fixed, Jittered, and Sporadic Release Times

- In many systems, we do not know exactly when each job will be released.
- The actual release time is not fixed.
- r_i^- : the earliest release time, r_i^+: the latest release time.
- Release time jitter:$[r_i^-, r_i^+]$;
- Sporadic jobs
  - Ex: the pilot may disengage the autopilot system at any time. The autopilot system change from cruise mode to standby mode. This job that execute to accomplish this mode change is sporadic job.
- Aperiodic jobs
- The release times of sporadic and aperiodic jobs are random variables.
- The model of the system gives the probability distribution A(*x*) of the release time of the job.

CPS LAB

Execution time

- A job J_i will execute for time e_i
  - This is the amount of time required to complete the execution of J_i when it executes alone and has all the resources it needs
  - Value of e_i depends upon complexity of the job and speed of the processor on which it is scheduled; may change for a variety of reasons:
    - Conditional branches
    - Cache memories and/or pipelines
    - Compression (e.g. MPEG video frames)
- Execution times fall into an interval $[e_i^-, e_i^+]$; assume that we know this interval for every hard real-time job, but not necessarily the actual e_i
  - Terminology: $(x, y]$ is an interval starting immediately after x, continuing up to and including y
- Often, we can validate a system using e_i for each job; we assume $e_i = e_i^+$ and ignore the interval lower bound
  - Inefficient, but safe bound on execution time

CPS LAB

3.3 Periodic task model

Period, Execution Times

- The *periodic task model* is a well-known deterministic workload model.
- The accuracy of the periodic task model decreases with the increasing jitter in release times and variations in execution times.
  - Ex: Inaccurate model of transmission of a variable bit-rate video.
- *Periodic task*, denoted by T_i, $(T_1, T_2, \ldots, T_n)$ is a sequence of jobs, $(J_{i,1}, J_{i,2} \ldots J_{i,k})$.
- *Period*, p_i, of the periodic task T_i is the minimum length of all time interval between release times of consecutive jobs in T_i.
- Execution time, e_i, is the maximum execution time of all the jobs in it.

Phases of Periodic Tasks

- The release time $r_{i,1}$ of the first job $J_{i,1}$ in each task T_i is called the *phase* of T_i.
- Φ_i is used to denote the phase of T_i, that is $\Phi_i = r_{i,1}$.
- Tasks are *in phase*, meaning that they have the same phase.
- H is used to denote the least common multiple of p_i, for $i = 1,2,3\ldots n$.
- A time interval of length H is called a *hyperperiod* of the periodic tasks.
- The number N of jobs in each hyperperiod is equal to $\sum_{i=1}^{n} \frac{H}{p_i}$.
  - ex: the length of a hyper period of three tasks with period 3, 4, and 10 is 60.
  - The total number N of jobs in the hyper period is 41.

CPS LAB

Utilization of tasks

- $u_i = e_i/p_i$ the utilization of the task T_i, is equal to the fraction of time a truly period task with period p_i and execution e_i keeps a processor busy.
- The *total utilization U* of all the tasks in the system is the sum of the utilizations of the individual tasks in it.
  - Ex: three periodic tasks, their execution times are 1, 1, and 3, and their periods are 3, 4, and 10, respectively.
  - Their utilizations are 1/3 = 0.33, ¼ = 0.25, 1/10 = 0.1 and the total utilization of the tasks is 0.88. These tasks keep a processor busy at most 88 percent of time.
- D_i is the relative deadline of task T_i .
  - Usually $D_i = p_i$, when jobs are released and become ready at the beginning of each period and complete by the end of the period.
  - D_i can have an arbitrary value and short than p_i.
  - Ex: a computation job is released but wait for input data transferred into memory
  - Ex: jobs has to be executed in a sequence.

Sporadic and Aperiodic Tasks

- A real-time system is required to respond to external events.
  - Ex: an operator's adjustment of sensitivity setting of a radar surveillance system.
- Jobs in each non-periodic task are similar and have the same statistical behavior and the same timing requirement.
  - A(x), inter-arrival times are identically distributed random variables with some probability distribution.
  - B(x), execution times are identically distributed random variables with some probability distribution
- *Sporadic Tasks* - Tasks containing jobs that are released at random time instants and have hard deadlines.
- *Aperiodic Tasks* – Tasks that the jobs in it have either soft deadlines or no deadlines.

3.4 Precedence constraints and data dependency

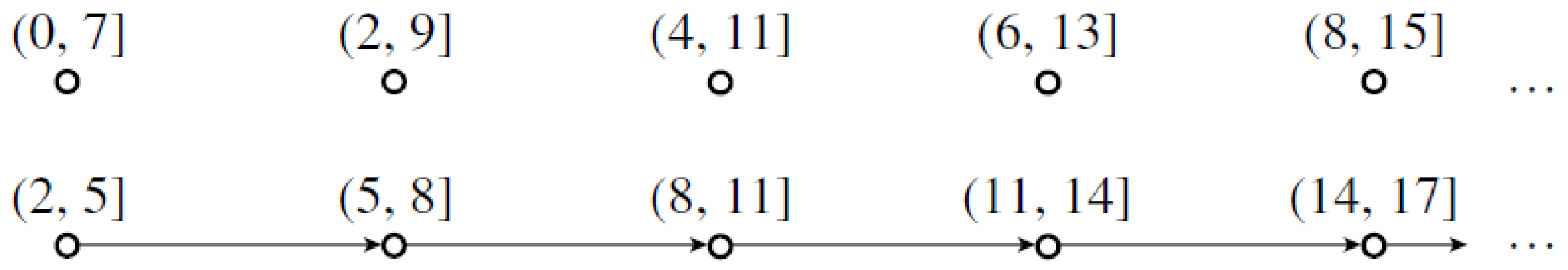
Precedence constraints

- Example:
  - Information server.
- $J_i < J_k$: A job J_i is a *predecessor* of another job J_k (and J_k is a *successor* of J_i) if J_k cannot begin execution until the execution of J_i completes.
- J_i is an *immediate predecessor* of J_k if there is no other job J_j such that $J_i<J_j<J_k$.
- Two jobs J_i and J_k are independent when neither $J_i<J_k$ nor $J_k<J_i$.
- *Precedence graph*: a directed graph G = (J, <) used to represent the precedence constraints among jobs in a set J.
- A *task graph* is an extended precedence graph to describe the application system.

CPS LAB

Example of task graphs

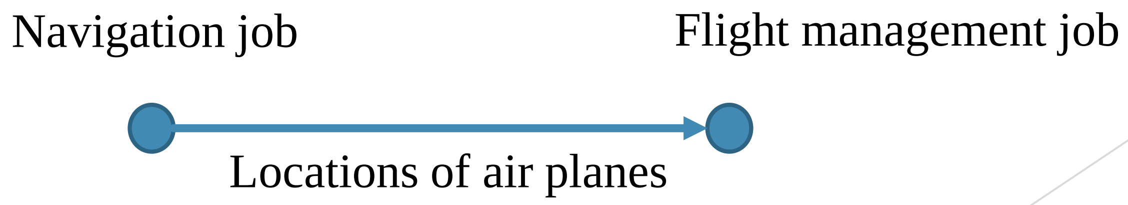
- Two periodic tasks.
- The one in the top row, has phase 0, period 2, and relative deadline 7.
- The one in the second row, has phase 2, period 3, and relative deadline 3.



CPS LAB

Data dependency

- Data dependency can not be captured by a precedence graph.
- In many real-time systems, jobs communicate via shared data.
  - Ex: in an avionics system, the navigation job updates the location of the air plane periodically.
  - Whenever the flight management job needs 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 explicitly by data-dependency edges among jobs.



End