## Real Time and Embedded Systems

- Revision to weighting of assessed coursework vs final examination
  - 20% of grade derived from assessed course work
    - 2 problems sets @ 4% each
    - 1 programming assignment @ 12%
  - 80% of grade derived from exam mark
- Other data
  - Prof. Sventek's office: S162
  - Dr. Perkins's office: S154



## A Reference Model of Real-Time Systems

- A good model permits us to focus on the important aspects of a system while ignoring the irrelevant properties/details
- Our reference model is characterized by:
  - 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
- Today, we will focus on the first two elements of the reference model – the models that describe the applications and resources

## A Reference Model of Real-Time Systems

- Processor – every job must have one or more processors in order to execute and make progress towards completion – e.g. computer, transmission link, disk, database server
- Each processor has a speed attribute – the rate of progress a job makes toward completion depends upon the speed of the processor upon which it executes
- Processor ***type*** – two processors are of the same type if they are functionally identical and can be used interchangeably – e.g. two transmission links with the same xmt rate between a sender/receiver pair, processors in a symmetrical multiprocessor system
- Resources – passive entities in the system upon which jobs depend – e.g. memory, sequence numbers, mutexes, database locks
- A resource does NOT have a speed attribute



## A Reference Model of Real-Time Systems

Processor/resource examples

- Computation job shares data with other computations
  - Shared data is guarded by a semaphore
  - Semaphore modelled as a resource
  - Job wanting to access the shared data must obtain the semaphore (lock it), use data, then release the semaphore
- Sliding window scheme to regulate message transmission
  - There is a maximum number of unacknowledged messages are allowed to be in transit; modelled as a set of valid sequence numbers
  - The set of valid sequence numbers moves forward as earlier messages are acknowledged; in order for a message to be transmitted, it must be assigned one of the valid sequence numbers
  - Model the transmission of a message as a job; this job executes when the message is being transmitted
  - This job needs the data link and a valid unit of the sequence number resource

## A Reference Model of Real-Time Systems

- We usually talk about ***reusable*** resources – i.e. they are not consumed during use
- If the system contains ρ resources, this means:
  - There are ρ types of serially reusable resources
  - There are one or more units of each type of resource
  - Each unit is used in a mutually exclusive manner
  - A job must obtain a unit of a needed resource and then release it
- A resource is ***plentiful*** if no job is ever prevented from executing by the unavailability of units of the resource – i.e. it never blocks when attempting to obtain a unit of a plentiful resource – e.g. obtaining the contents of a read-only file



## A Reference Model of Real-Time Systems

- Temporal parameters of a RT workload
  - Many parameters of hard RT jobs and tasks are known at all times – otherwise, we cannot ensure that the system meets its hard RT requirements
    - The number of hard RT tasks or jobs – a hard RT system may operate in different modes – the number of tasks/jobs is know for each mode – e.g. autopilot system is changed to standby
    - Each job $J_i$ is characterized by its temporal (timing constraints), functional (intrinsic properties of the job), resource (needed resources), and interconnection parameters (interdependency with other jobs)

# A Reference Model of Real-Time Systems

- Temporal concepts
  - $r_i$ – release time of $J_i$
  - $d_i$ – absolute deadline of $J_i$
  - $D_i$ – relative deadline of $J_i$
  - $(r_i, d_i]$ – feasible interval for $J_i$
  - Often do not know exactly when a job is released, only that $r_i$ is in a range $[r_i^-, r_i^+]$ – this is known as ***release time jitter***
  - 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



# A Reference Model of Real-Time Systems

- Nearly every real time system is required to respond to external events which occur at random instants of time
- The jobs resulting from these events are called sporadic or aperiodic jobs because they are released at random instants of time
- The release times for sporadic/aperiodic jobs are random variables; in the model, we use a probability distribution A(t) for the probability of t being the release time of a job; alternatively, when discussing a stream of similar sporadic/aperiodic jobs, it is the probability distribution for interrelease time
- A(x) gives us the probability that the release time of the job is at or earlier than x (or the interrelease time between successive jobs in the stream is less than or equal to x)
- Sometimes we use the terms arrival time (interarrival time) due to its common use in queueing theory. A sporadic/aperiodic jobs arrives when it is released.

## A Reference Model of Real-Time Systems

- Execution time
  - $e_i$ is the execution time for $J_i$ – i.e. the amount of time required to complete the execution of $J_i$ when it executes alone and has all the resources it requires.
  - Value depends upon the complexity of the job and the speed of the processor upon which it is scheduled
  - Execution time may vary for a variety of reasons
    - Conditional branches
    - Cache memories and/or pipelines
    - Compression (e.g. MPEG video frames)
  - As for release time, usually we know $e_i$ is in the range $[e_i^-, e_i^+]$; we usually assume that we know this range for every hard RT job
  - Often, we can validate a system by knowing $e_i^+$ for each job; therefore, $e_i$ often implies the maximum execution time



## A Reference Model of Real-Time Systems

- Periodic task model
  - Each computation or data transmission that is executed repeatedly at regular or semi-regular time intervals is modelled as a periodic task
  - Each periodic task $T_i$ is a sequence of jobs
  - The period $p_i$ of $T_i$ is the minimum length of all time intervals between release times of consecutive jobs
  - The execution time $e_i$ of $T_i$ is the maximum of all jobs in the periodic task.
  - The period and execution time of every periodic task in the system are known at all times.
  - The accuracy of the periodic task model decreases with increasing release jitter and variations in execution times

## A Reference Model of Real-Time Systems

- Periodic task model
  - Individual jobs in $T_i$ are referred to as $J_{i,1}$, $J_{i,2}$, …
  - The release time $r_{i,1}$ of $J_{i,1}$ in each task $T_i$ is called the phase of $T_i$, $\varphi_i$
  - The hyperperiod H of a set of periodic tasks is the least common multiple of $p_i$ for I = 1 … N
  - The ratio $u_i = e_i/p_i$ is the utilization of task $T_i$ – i.e. the fraction of time a truly periodic task with period $p_i$ and execution time $e_i$ keeps a processor busy
  - Total utilization $U = \Sigma u_i$, where the sum is over all periodic tasks in the system
  - The relative deadline, $D_i$, is often simply the period, $p_i$



## A Reference Model of Real-Time Systems

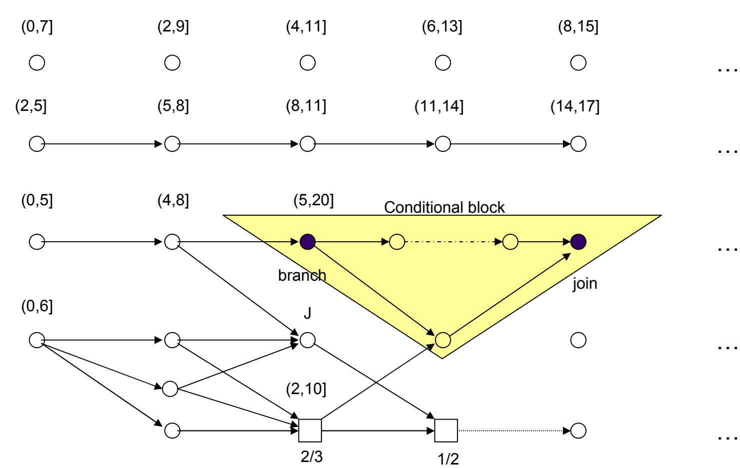
- Sporadic/aperiodic tasks
  - Each sporadic/aperiodic task is a stream of sporadic/aperiodic jobs
  - The interarrival times between consecutive jobs in such a task may vary widely and, in particular, can be arbitrarily small
  - The interarrival times of consecutive jobs are identically distributed random variables with some probability distribution A(x)
  - Similarly, the execution times of jobs are identically distributed random variables with some probability distribution B(x)
  - A task is sporadic if its jobs have hard deadlines
  - A task is aperiodic if its jobs have either soft deadlines or no deadlines

# A Reference Model of Real-Time Systems

- Precedence constraints and data dependencies
  - Jobs are said to have ***precedence constraints*** if they are constrained to execute in a particular order; otherwise they are independent
  - 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 – $J_i < J_k$
  - $J_i$ is an ***immediate predecessor*** of $J_k$ if $J_i < J_k$ and there is no other job $J_j$ such that $J_i < J_j < J_k$
  - $J_i$ and $J_k$ are independent when neither $J_i < J_k$ nor $J_k < J_i$
  - Represent the precedence constraints among jobs in a set **J** using a directed graph G = (**J**, <); each vertex is labelled by the name of the job represented; a directed edge goes from $J_i$ to $J_k$ if $J_i$ is an immediate predecessor of $J_k$



# A Reference Model of Real-Time Systems



## A Reference Model of Real-Time Systems

- Task graphs
  - Jobs represented by circles and squares
  - Directed edges represented by arrows
  - AND/OR precedence constraints
    - Normally a job must wait for the completion of all immediate predecessors – termed an AND constraint
    - An OR constraint indicates that a job may begin after its release time if only some of the immediate predecessors have completed – k-out-of-l – shown as boxes in the task graph
    - The in-type of a node in the graph indicates whether it is an AND or an OR node



## A Reference Model of Real-Time Systems

- Task graphs
  - Conditional branches
    - Normally, all the immediate successors of a job must be executed; an outgoing edge from every vertex expresses an AND constraint
    - Conditional branches can be represented by outgoing edges with an OR constraint – indicates that only 1 of the immediate successors is to be executed – nodes represented by filled-in circles in a task graph
    - Conditional branch is the subgraph from the node with the OR conditional constraint to the corresponding join node
  - Pipeline relationship
    - Need a way to represent a pair of producer/consumer jobs
    - Represented in task graphs with a dotted edge

## A Reference Model of Real-Time Systems

- Functional parameters
  - Preemptivity of jobs
  - Importance or criticality of jobs
  - Optional jobs or portions of jobs
  - Laxity type and function (usefulness)
- Resource parameters
  - Preemptivity of resources



## A Reference Model of Real-Time Systems

- Scheduling
  - Jobs are scheduled and allocated resources according to a chosen set of scheduling algorithms and resource access-control protocols; a scheduler implements these algorithms
  - A scheduler specifically assigns jobs to processors
  - A schedule is an assignment of all jobs in the system on the available processors.
  - A ***valid schedule*** satisfies the following conditions:
    - Every processor is assigned to at most one job at any time
    - Every job is assigned at most one processor at any time
    - No job is scheduled before its release time
    - The total amount of processor time assigned to every job is equal to its maximum or actual execution time
    - All the precedence and resource usage constraints are satisfied

# A Reference Model of Real-Time Systems

- Scheduling
  - A valid schedule is a ***feasible schedule*** if every job meets its timing constraints.
  - A hard real time scheduling algorithm is ***optimal*** if the algorithm always produces a feasible schedule if the given set of jobs has feasible schedules.
  - lateness = completion time – deadline
    tardiness = max[0, lateness]
  - Miss rate – the percentage of jobs that are executed but completed too late
  - Loss rate – the percentage of jobs that are not executed at all