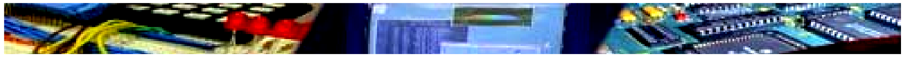
Scheduling Algorithm and Analysis

Model and Cyclic Scheduling

(Module 27)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014

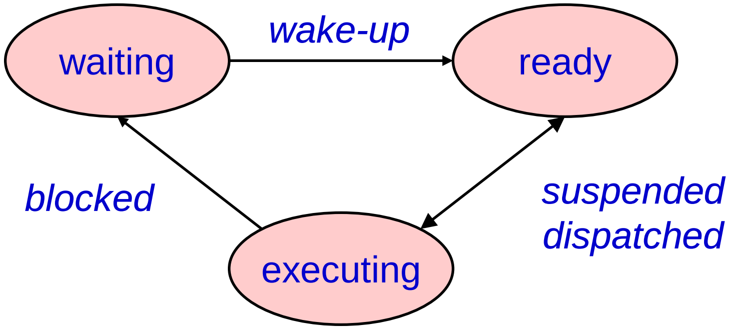


Task Scheduling

- **Schedule: to determine which task is assigned to a processor at any time**
  - order of execution
  - meet deadlines, fast response time, utilize resource effectively
- **Need an algorithm to generate a schedule**
  - optimal scheduling algorithm: always find a feasible schedule if and only if a feasible schedule exists
- **Scheduler or dispatcher: the mechanism to implement a schedule**
- **Misconcept:**
  - RTOS will schedule tasks to meet task deadlines
  - A good schedule will reduce CPU load

Task Functional Parameters

- **Preemptivity: suspend the executing job and switch to the other one**
  - should a job (or a portion of job) be preemptable
  - context switch: save the current process status (PC, registers, etc.) and initiate a ready job
  - transmit a UDP package, write a block of data to disk, a busy waiting loop
- **Preemptivity of resources: concurrent use of resources or critical section**
  - lock, semaphore, disable interrupts
- **How can a context switch be triggered?**
  - Assume you want to preempt an
    - executing job -- why
    - a higher priority job arrives
    - run out the time quantum



Event- and Time-Triggered Systems

Time-triggered control system

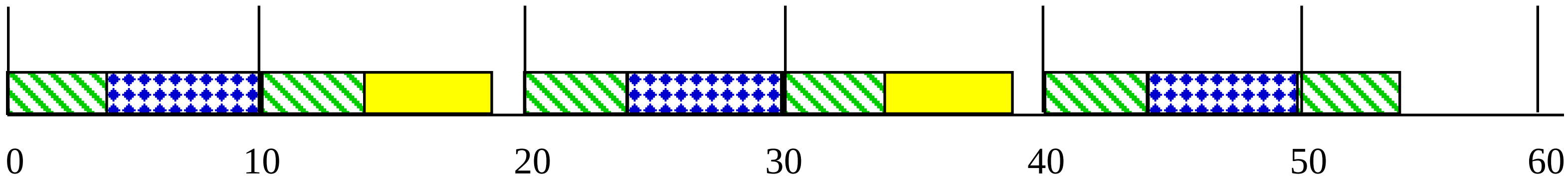
- All activities are carried out at certain points in time know a priori at design time (based on a globally synchronized time base)
  - Transmission of messages
  - Task execution
  - Monitoring of external states
- All nodes have a common notion of time

Event-triggered control system

- All activities are carried out in response to events external to the system
  - Reception of a message
  - Termination of a task
  - External interrup

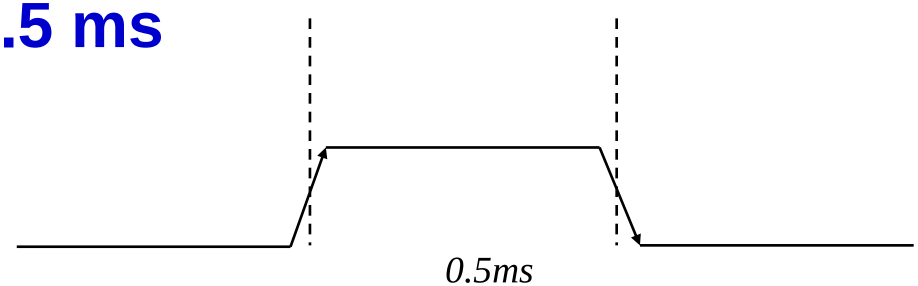
Major and Minor Cycle Model

- **Time is divided into equal-sized frame**
  - minor cycle = length of frame
  - Major cycle = length of schedule = k * minor_cycle
- **An example: A=(10,4) B=(20,6) C=(30,5)**
  - major cycle=60, minor cycle=10
  - scheduling string AB_AC_AB_AC_AB_A_
- **Jobs must be done within a minor cycle**
  - limit timing error to one frame
  - suspend and resume as background, continue, or abort if overrun



An Example

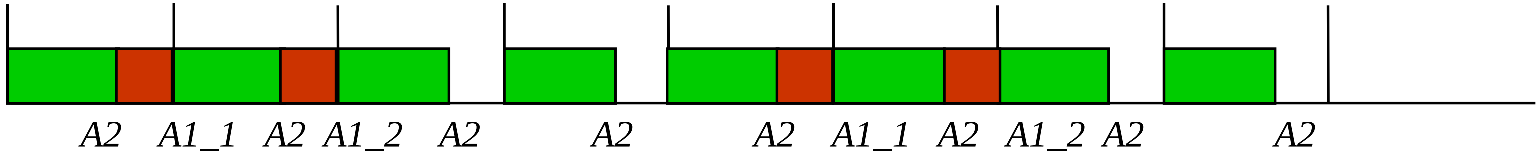
- A1 must be done at least every 10ms, and takes 1ms
- A2 must be completed with 5ms when E occurs and takes 2 ms
- E must be detected by polling and is detectable for at least 0.5 ms



- E would not occur twice within 50 ms
- polling of E takes 0 overhead

Major/Minor Cyclic Scheduling

- **There should be a periodic polling action for E**
  - Assume a timer of 0.5ms to activate polling operation and no polling overhead
- **Should be an interval of 2ms to execute A2 for an arbitrary 5ms interval**
  - May detect E in the first frame and execute A2 in the second frame $\Rightarrow$ period=2.5ms
  - A2 takes 2ms if E, otherwise is 0 $\Rightarrow$ WCET=2ms
- **Should be an interval of 1ms to execute A1 for an arbitrary 10ms interval**
  - Period= 10ms, WCET= 1ms
  - Since 2ms + 1ms > 2.5ms, we will divide A1 into two parts of 0.5ms



Summary of Cyclic Schedule

Pros

- simple, table-driven, easy to validate (knows what is doing at any moment)
- fit well for harmonic periods and small system variations
- static schedule $\Rightarrow$ deterministic, static resource allocation, no preemption
- small jitter
- no scheduling anomalies

Cons

- difficult to change (need to re-schedule all tasks)
- fixed released times for the set of tasks
- difficult to deal with different temporal dependencies
- schedule algorithm may get complex (NP-hard)
- doesn’t support aperiodic and sporadic tasks efficiently