19Z604 Embedded Systems

Dr.N.Arulanand,
Professor,
Dept of CSE,
PSG College of Technology

Real Time Operating Systems

REAL TIME OPERATING SYSTEMS: Real-Time Concepts - Task Management - Task Scheduling - Classification of Scheduling Algorithms - Clock Driven Scheduling - Event Driven Scheduling - Resource Sharing - Priority Inheritance Protocol - Priority Ceiling Protocol Commercial RTOS – VxWorks

What is Real Time ?

- A system is called a real-time system, when we need quantitative expression of time (i.e. real-time) to describe the behavior of the system

- A chemical plant, whose part behavior description is - when temperature of the reaction chamber attains certain predetermined temperature value, say 250^{o}C, the system automatically switches off the heater within say 30 milliseconds - is clearly a real-time system.

Applications of Real-Time

- Industrial Applications
  - Chemical Plant Control
  - Automated Car Assembly Plant
- Medical
  - Robot Used in Recovery o Displaced Radioactive Material
- Automotive &Transportation
  - Multi-Point Fuel Injection (MPFI) System : An automotive engine control system, that controls the rate of fuel injection and allows the engine to operate at its optimal efficiency
- Defense Applications
  - Missile Guidance System

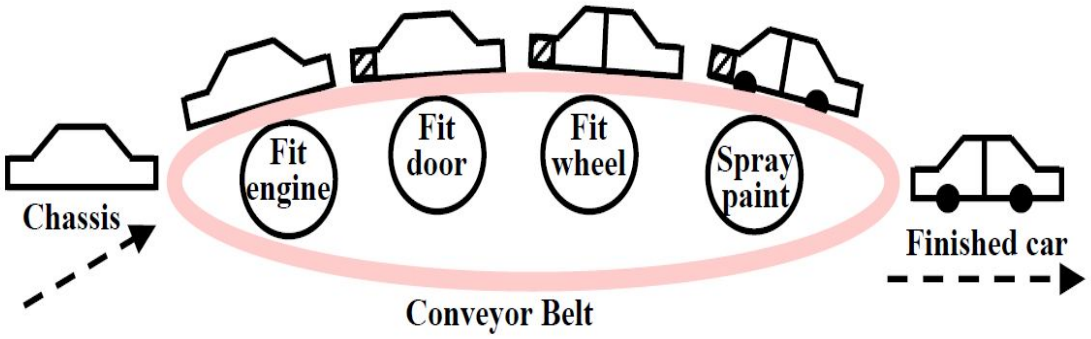


Fig. 28.1 Schematic Representation of an Automated Car Assembly Plant

Characteristics of Real-time system

- **Time constraints**
  - Every real-time task is associated with some time constraints. One form of time constraints that is very common is deadlines associated with tasks.
- **New Correctness Criterion**
  - In real-time systems, correctness implies not only logical correctness of the results, but the time at which the results are produced
  - A logically correct result produced after the deadline would be considered as an incorrect result.
- **Safety-Criticality**
  - For traditional non-real-time systems safety and reliability are independent issues. However, in many real-time systems these two issues are intricately bound together making them safety-critical
  - a safe system is one that does not cause any damage even when it fails.
  - A reliable system on the other hand, is one that can operate for long durations of time without exhibiting any failures.

Hard Real-time System

- A hard real-time task is one that is constrained to produce its results within certain predefined time bounds.
- The system is considered to have failed whenever any of its hard real-time tasks does not produce its required results before the specified time bound.

Soft Real-Time System

- Soft real-time tasks also have time bounds associated with them.
- However, unlike hard real-time tasks, the timing constraints on soft real-time tasks are not expressed as absolute values.
- Instead, the constraints are expressed either in terms of the average response times required.

Basic Terminologies

- **Task Instance**
  - Each time an event occurs, it triggers **the task** that handles this event to run
  - Real-time tasks therefore normally recur a large number of times at different instants of time depending on the event occurrence times.
  - It is possible that real-time tasks recur at random instants.
  - However, most real-time tasks recur with certain fixed periods.

Types of Task

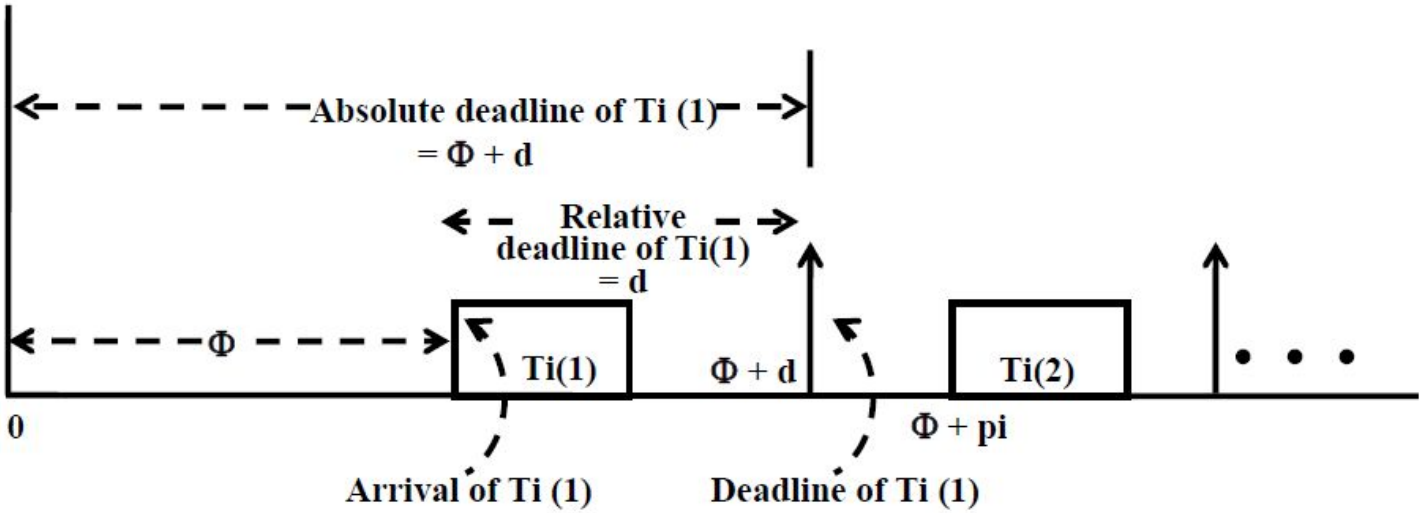
- **Periodic Task:**
  - A periodic task is one that repeats after a certain fixed time interval.
- $T_i = (e_i, pi, d_i)$ Highly critical
- **Sporadic Task:**
  - A sporadic task is one that recurs at random instants. A sporadic task Ti can be is represented by a three tuple: $T_i = (e_i, g_i, d_i)$

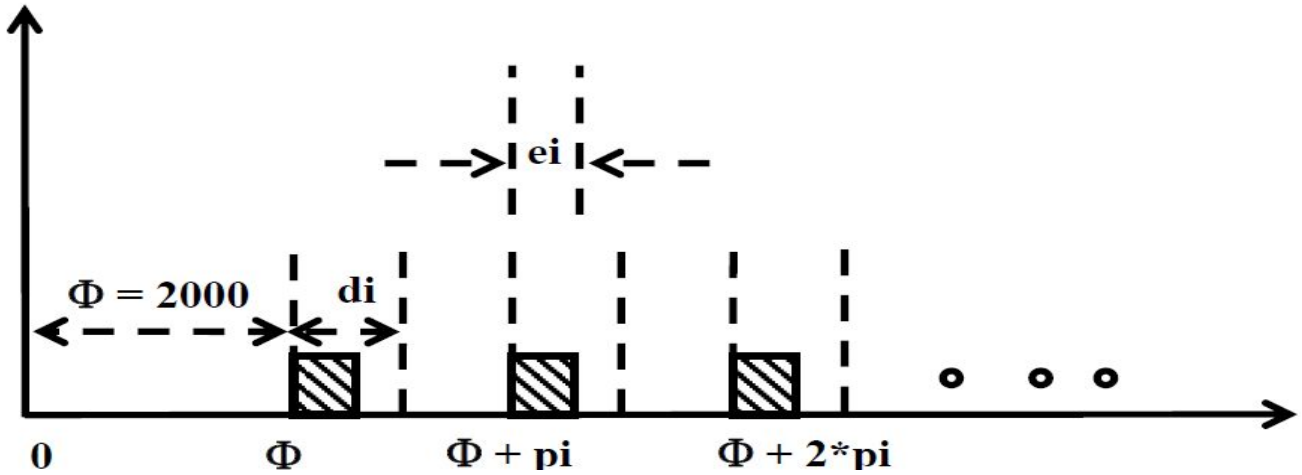
  Critical to Normal

  where e^i is the worst case execution time of an instance of the task, g^i denotes the minimum separation between two consecutive instances of the task, d^i is the relative deadline.

- **Aperiodic Task**
  - An aperiodic task can arise at random instants . the minimum separation gi between two consecutive instances can be 0
  - Low priority

Task terminologies





Notation

The 4-tuple $T_i = (\phi_i, p_i, e_i, D_i)$ refers to a periodic task T_i with phase ϕ_i, period p_i, execution time e_i, and relative deadline D_i

- Default phase of T_i is $\phi_i = 0$, default relative deadline is the period $D_i = p_i$
- Omit elements of the tuple that have default values
- Examples:

$T_1 = (1, 10, 3, 6) \Rightarrow \quad \phi_1 = 1 \quad p_1 = 10 \quad e_1 = 3 \quad D_1 = 6$

$J_{1,1}$ released at 1, deadline 7
$J_{1,2}$ released at 11, deadline 17
…

$T_2 = (10, 3, 6) \quad \Rightarrow \quad \phi_2 = 0 \quad p_2 = 10 \quad e_2 = 3 \quad D_2 = 6$

$J_{1,1}$ released at 0, deadline 6
$J_{1,2}$ released at 10, deadline 16
…

$T_3 = (10, 3) \quad \Rightarrow \quad \phi_3 = 0 \quad p_3 = 10 \quad e_3 = 3 \quad D_3 = 10$

$J_{1,1}$ released at 0, deadline 10
$J_{1,2}$ released at 10, deadline 20
…

Types of Schedulers

- 1. Clock-driven Schedulers : based on clock interrupt.
  1.1 Table driven
  1.2 Cyclic Scheduler
- 2. Event-driven Schedulers
  2.1 RMA (Rate Monotonic Analysis)
  2.2 EDF (Earliest Deadline First)
  2.3 DMA (Deadline Monotonic Analysis)
- 3. Hybrid
  – Round-robin

Clock-driven Schedulers

- Clock-driven schedulers make their scheduling decisions regarding which task to run next only at the clock interrupt points.
- The scheduling points are determined by timer interrupts
- Handle only periodic tasks.
- They are simple and efficient.
- Easy to implement
- A.k.a. Off-line Schedulers because these schedulers fix the schedule before the system starts to run.

1.1 Table Driven Scheduler

- Pre-compute which task would run and when.
- Store this schedule in a table

- Storage capacity

Task	*Start time in millisecs*
T_1	*0*
T_2	*3*
T_3	*10*
T_4	*12*
T_5	*17*

Q: what would be the size of the schedule table ?

if a set ST = {Ti} of n tasks is to be scheduled, then the entries in the table will replicate themselves after LCM (p1, p2… ,pn) time units, where p1, p2 …pn are the periods of T1, T2…Tn

1.1 Table Driven Scheduler

- Example:
  T1 = (e1=5 msecs, p1=24 msecs),
  T2 = (e2=10 msecs, p2=48 msecs)
  T3= (e3=20 msecs, p3=60 msecs)

  LCM (24,48,60) = 240 msec
- **Major Cycle:** The major cycle of a set of tasks ST = {T1, T2, … , Tn} is LCM ({p1, p2, … , pn})
  where p^1, p^2, …, p^n are the periods of T^1, T^2, ..., T^n

- Table Size = LCM of {P1,P2,P3}

Task	Start Time (in ms)
T1	0
T2	5
T3	15
T1	35
T1	48
T2	53
	… … 240 msec

Homework

- Complete the table for the above example

1.1 Table Driven Scheduler

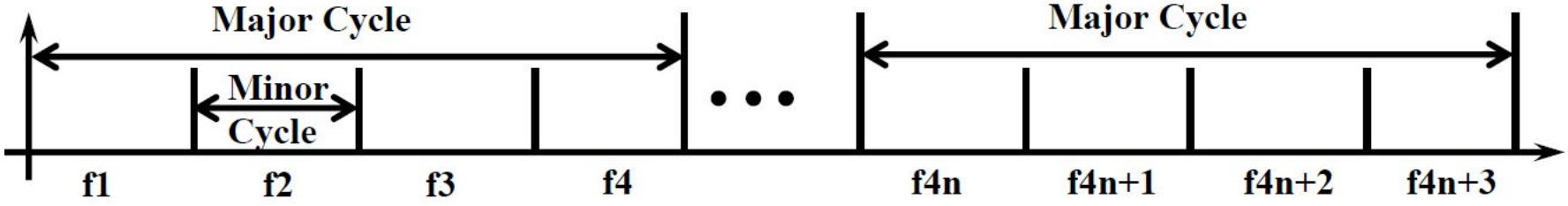
- All parameters about jobs (execution time/deadline) known in advance
- Schedule can be computed offline or at some regular time instances.
- Minimal runtime overhead.
- Not suitable for many applications.

1.2 Cyclic Scheduler

- Simple, Efficient, Easy to program
- Used in small Embedded Applications
- Cyclic scheduler repeats a pre-computed schedule. Pre-computed schedule need to be stored for only one major cycle.
- Major and Minor cycle
  - The major cycles in divided into one or more minor cycles
  - Minor cycle is aka Frame
  - Scheduling points occur at frame boundaries
  - Frame boundaries are interrupts generated by a periodic timer.

1.2 Cyclic Scheduler

- Cyclic schedulers are very popular and are being extensively used in the industry.
- small embedded applications
- A cyclic scheduler repeats a pre-computed schedule.
- The pre-computed schedule needs to be stored only for one major cycle
- The major cycle has been divided into minor cycles (frames)Each minor cycle is also sometimes called a frame.
- The scheduling points of a cyclic scheduler occur at frame boundaries. This means that a task can start executing only at the beginning of a frame.
- The frame boundaries are defined through the interrupts generated by a periodic timer.



Cyclic Scheduler

- Schedule task table
- Size of the frame is a important design parameter. Need to chosen carefully.

- Criteria for selecting Frame Size

  Constraint 1: Minimum Context Switching

  Constraint 2: Minimization of Table Size

  Constraint 3: Satisfaction of Task Deadline

  If all the constraints are satisfied, then the frame size can be selected

Cyclic Scheduler

- Minimum Context Switching
  - **F >= Max{ei}**; F is frame size, ei is the execution time of task Ti
  - Min. no of context switches during task execution. i.e., a task instance must complete within the assigned frame.
  - If the task doesn't complete, it needs to be suspended and restarted in a later frame.
  - Context switching involves processing overhead
  - Set the selected frame size larger than the execution time.

Cyclic Scheduler

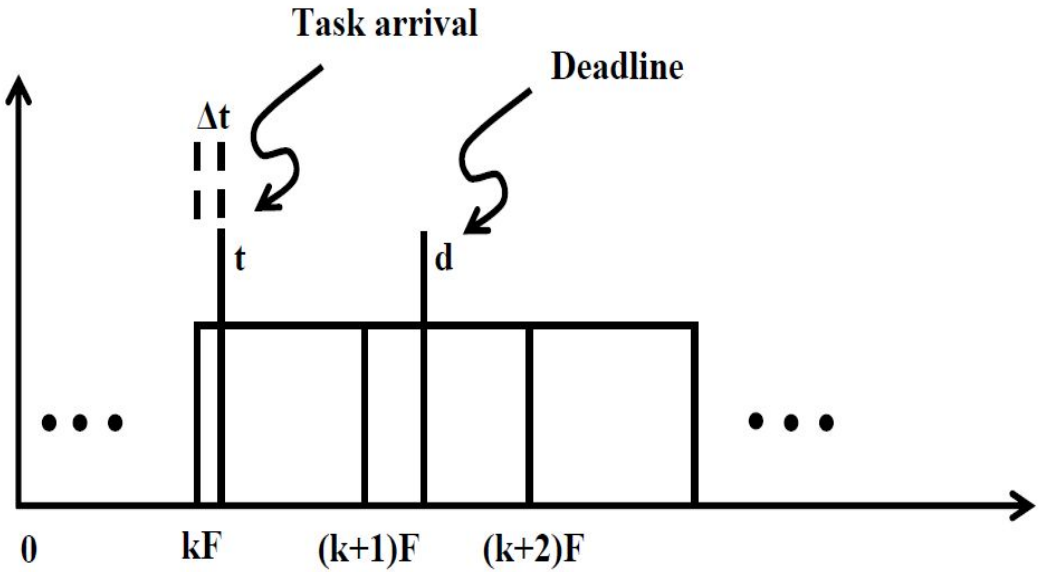
- Minimization of Table Size
  - No of entries in the schedule table should be minimum. (for minimum storage, as this is used for smaller embedded applications)
  - Entries can be minimised, if the Minor cycle is squarely divides the major cycle. **(no fractional minor cycles)**
  - **M – major cycle**
  - **F – frame/minor cycle**
  - Formula: If **Floor(M/F) is equal to M/F**

  (or) M mod F = 0

Cyclic Scheduler

- Satisfaction of Task Deadline
  - This is needed to meet the task deadlines.
  - Between arrival of a task and its deadline, there must exist at least one full frame.
  - **For every Ti,**

    2F – gcd(F,pi) <= di



Cyclic Scheduler

Exercise 1: A cyclic scheduler is to be used to run the following set of periodic tasks on a uniprocessor:

T1 = (e1=1ms, p1=4ms)
T2 = (e2=1ms, p2=5ms)
T3 = (e3=1ms, p3=20ms),
T4 = (e4=2ms, p4=20ms).

Select an appropriate frame size ?

Constraint 1: F >= max (ei)

F >= max(e1,e2,e3,e4)
F > = max(1,1,1,2)
F >= 2

Constraint 2: M mod F = 0

M = LCM(p1,p2,p3,p4)
M = LCM(4,5,20,20)
M = 20
20 mod 2 = 0
20 mod 4 = 0
....
....
F = 2,4,5,10,20

Cyclic Scheduler

Constraint 3: 2F – gcd(F, pi) <= di

Pi = di (if deadline is not given in the problem)

For F = 2 and Task 1:Pi = 4

2*2 – GCD(2, 4) <= 4

4 – 2 <= 4 (condition satisfied)

2 <= 4

For F = 2 and Task 2: Pi = 5

2*2 – GCD(2, 5) <= 5

4 – 1 <= 5 (condition satisfied)

For F = 2 and Task 3: Pi = 20

2*2 – GCD(2, 20) <= 20

4 – 2 <= 20 (condition satisfied)

For F=2 and Task 4: Pi = 20

2*2 – GCD(2, 20) <= 20

4 – 2 <= 20 (condition satisfied)

Thus constraint 3 is satisfied by all tasks for frame size 2. So, Frame size 2 satisfies all the three constraints. Hence frame 2 is a feasible frame

Cyclic Scheduler

Constraint 3: 2F – gcd(F, pi) <= di

Pi = di (if deadline is not given in the problem)

For F = 4 and Task 1:Pi = 4

2*4 – GCD(4, 4) <= 4

8 – 4 <= 4 (condition satisfied)

For F = 4 and Task 2: Pi = 5

8 – GCD(4,5) < = 5

Thus constraint 3 is NOT satisfied for frame size 4.

Hence frame 4 is NOT a suitable frame

Cyclic Scheduler

- Exercise 1: A cyclic scheduler is to be used to run the following set of periodic tasks on a uniprocessor:

  T1 = (e1=1ms, p1=4ms)

  T2 = (e2=1ms, p2=5ms)

  T3 = (e3=1ms, p3=20ms)

  T4 = (e4=2ms, p4=20ms).

Select an appropriate frame size ?

Static Cyclic Scheduler

A cyclic real-time scheduler is to be used to schedule three periodic tasks T1, T2, and T3 with the following characteristics:

Task	**Phase** mSec	**Execution Time** mSec	**Relative Deadline** mSec	**Period** mSec
T_1	0	20	100	100
T_2	0	20	80	80
T_3	0	30	150	150

Suggest a suitable frame size that can be used. Show all intermediate steps in your calculations.

Static Cyclic Scheduler

Consider the following set of three independent real-time periodic tasks.

Task	**Start Time** mSec	**Processing Time** mSec	**Period** mSec	**Deadline** mSec
T_1	20	25	150	100
T_2	40	10	50	30
T_3	60	50	200	150

Suppose a cyclic scheduler is to be used to schedule the task set. What is the major cycle of the task set? Suggest a suitable frame size and provide a feasible schedule (task to frame assignment for a major cycle) for the task set.

Cyclic Scheduler

Consider the following set of periodic real-time tasks to be scheduled by a cyclic scheduler:

T1 = (e1=1ms, p1=4ms),

T2 = (e2=2ms, p2=5ms)

T3 = (e3=5ms, p3=20ms)

Determine a suitable frame size for the task set ?

Cyclic Scheduler

Exercise 3: Consider the following set of periodic real-time tasks to be scheduled by a cyclic scheduler:

T1 = (e1=1ms, p1=4ms),

T2 = (e2=2ms, p2=5ms),

T3 = (e3=5ms, p3=20ms).

Determine a suitable frame size for the task set ?

Clue:

Split into sub-tasks and find the frame

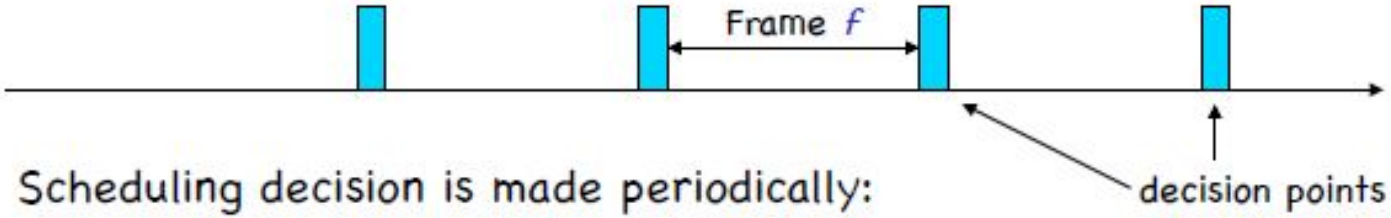
T3.1 = (1,20)

T3.2 = (2,20)

T3.3 = (2,20)

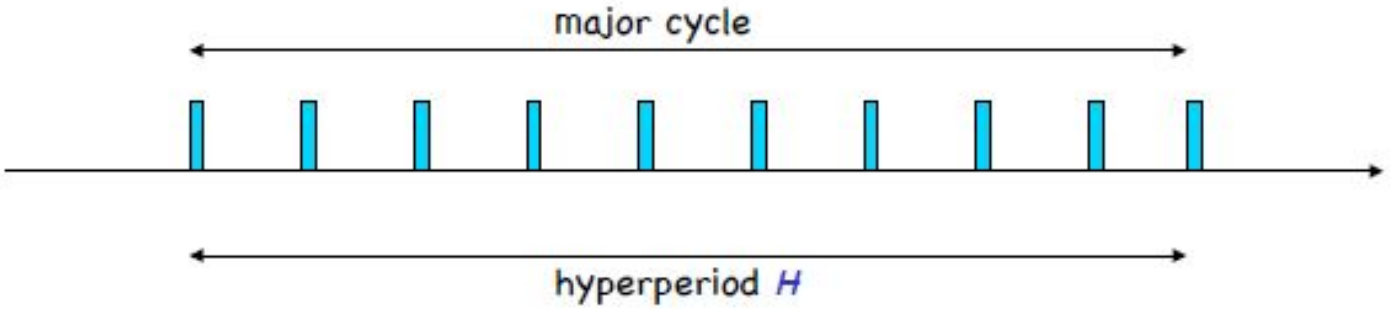
Cyclic Schedules: General Structure

- Scheduling decision is made periodically:



- Scheduling decision is made periodically:
  - choose which job to execute
  - perform monitoring and enforcement operations

- Major Cycle: Frames in a hyperperiod.



Frame Size Constraints

- Frames must be sufficiently long so that every job can start and complete within a single frame:

$$(1) \qquad f \geq \max(e_i)$$

- The hyperperiod must have an integer number of frames:

$$(2) \qquad f \mid H \qquad (f \text{"}\textit{divides}\text{"} H)$$

- For monitoring purposes, frames must be sufficiently small that between release time and deadline of every job there is at least one frame:

$$2f - (t'-t) \leq D_i$$

$$t'-t \geq \gcd(p_i, f)$$

$$(3) \qquad 2f - \gcd(p_i, f) \leq D_i$$

Slicing and Scheduling Blocks

- Slicing

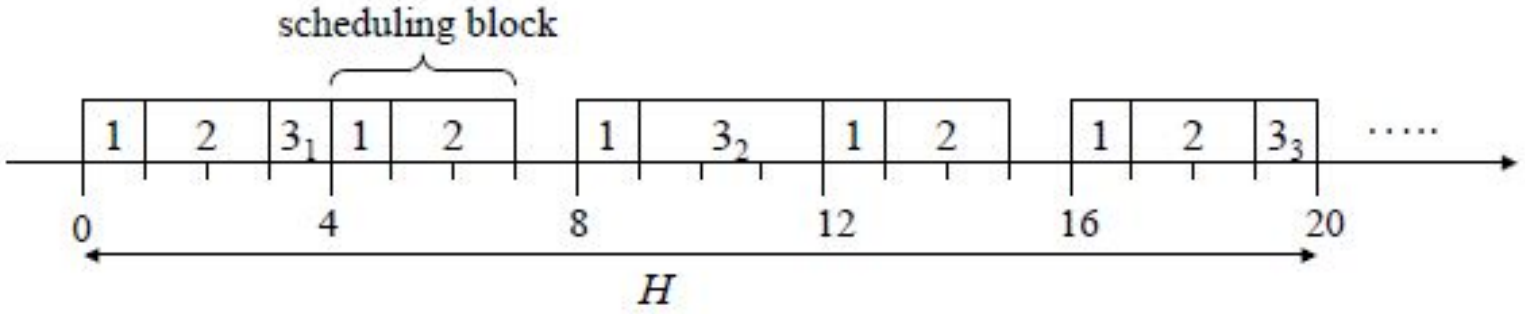
	p_i	e_i	D_i
T_1	4,	1,	4
T_2	5,	2,	5
T_3	20,	5,	20

$$\left.\begin{array}{lll} (1) & \Rightarrow & f \geq 5 \\ (3) & \Rightarrow & f \leq 4 \end{array}\right\} ?!$$

slice T_3

	p_i	e_i	D_i
T_1	4,	1,	4
T_2	5,	2,	5
T_{31}	20,	1,	20
T_{32}	20,	3,	20
T_{33}	20,	1,	20

$$\left.\begin{array}{lll} (1) & \Rightarrow & f \geq 3 \\ (3) & \Rightarrow & f \leq 4 \end{array}\right\} f = 4$$



Static cyclic scheduling: + and –

- Deterministic: predictable (+)
- Easy to implement (+)
- Inflexible (-)
  - Difficult to modify, e.g adding another task
  - Difficult to handle external events
- The table can be huge (-)
  - Huge memory-usage
  - Difficult to construct the time table

Example: shortest repeating cycle

- The LCM determines the size of the time table
  - LCM =50ms for tasks with periods: 5ms, 10ms and 25ms
  - LCM =7*13*23=2093 ms for tasks with periods: 7ms, 13ms and 23ms (very much bigger)
- So if possible, manipulate the periods so that they are multiples of each other
  - Easier to find a feasible schedule and
  - Reduce the size of the static schedule, thus less memory usage

Real Time System Pitfalls

Real time System Pitfalls -1: Patriot Missile

During 1991 Gulf war, an Iraqi Scud Missile hit an American army barrack killing many soldiers.
Actually an American patriot missile could not track and intercept the scud missile.
Reason:
A software bug in patriotic missile

When the system turned on then it measure the time in 100 ms. Intervals.
This value is multiplied by 10 to obtain second.
The calculation was performed in 24 bit floating register.
So, value of 1/10 th second is truncated in 24 bits
where as exact value is 0.00011 00110 01100 11001 1001100…
So, if value was truncated then en error occurred. (3.4 ms per hour)
The patriot missile was switched on for about 100 hours, so the accumulated error was 0.34 second.
Scud travels at speed of 1,676 mps, i.e. more then half a kilometer in 0.34 second.
So, patriot could not intercept the scud.

Real time System Pitfalls -2: Lockheed Martin

In 1998 Lockheed Martin Titan 4 booster carrying a $1 billion LockMart Vortex Class Spy satellite pitched sideways and exploded 40 seconds after liftoff cape Canaveral, Fla.

Reason:
Fried wiring that had not been inspected. The guidance system remain without power for fraction of second.

LOCKHEED MARTIN

The Ariane 5 Satellite Launch Rocket

Rocket self destructed in 4 June -1996.
Exactly after 40 second of lift off at an attitude of 3700 meters, the launcher exploded and became a ball of fire.
Cost: $500 Million USD
Reason:
Bad floating-point exception handling.
A 64 bit floating no is converted into 16 bit signed value.
During one calculation the converted value was more then of 16 bit.

This error is know as "**500 million dollar software error**"