Simple Scheduling

Slide 1

- Today we are going to cover two approaches to Scheduling.
  - Cyclic Executives
  - Simple priority based schedules
- As we get further in the course the scheduling theory will become more complicated to accommodate more realistic models.

Task types

We can divide tasks into two classes:

Slide 2

- *Sporadic* or aperiodic, that is tasks that arise because of some external stimulus in the environment (maybe using interrupts).
- *Periodic* Tasks which have to be repeated regularly over and over again.

Release Times, Deadlines and Timing Constraints

Slide 3

- The *Release time* of a task is the instant of time at which the job becomes available for execution.
- The *deadline* of a task is the instant of time by which its execution is required to be completed.
- The *Response time* of a task is the completion time (often measured relative to the release time) of the task.
- Any constraint, such as deadlines etc. , imposed on the timing behaviour of a job is called a *Timing Constraints*.

Task types continued

Slide 4

Both Sporadic and Periodic tasks have deadlines associated with them.

- *Periodic* tasks have a latest by when they have to executed each period.
- *Sporadic* given some external stimulus, there is a deadline by which the task must respond.

Converting Sporadic to periodic tasks

Slide 5

- There is very simple method for converting an aperiodic task to a periodic task simply turn it into a periodic task.
- Such a scheme can have a large overhead which we will analyse later.
- For the moment we shall assume there are no sporadic tasks.

Tasks and Processes

Slide 6

- We will implement our task with processes.
- These will be traditional OS type processes.
- For the moment we are going to ignore problems like context switches.
- We will also assume that we have complete control over when processes are executed. There will be no other processes to take priority or to take control.

Task Assumptions

For the moment we shall assume a limited task model.

Slide 7

- We have a fixed set of processes.
- All processes are periodic with known periods.
- The processes are completely independent of each other.
- All system overheads, context switching time etc are ignored.
- All processes have a deadline equal to their period.
- All processes have a fixed worst-case execution time.
- All deadlines will assumed to be hard.

Later we will be able to relax some of these assumptions.

Static and Dynamic Scheduling

There are two ways to schedule a set of tasks:

Slide 8

- *Statically* We compute a schedule before execution as part of the system specification.
- *Dynamically* We compute a schedule on the fly , that is as we go along.

Most of the research has been on static scheduling. Mainly because correct dynamic scheduling is too hard.

The Cyclic Executive Approach

Slide 9

Given a fixed set of purely periodic processes it is possible to lay out a complete schedule. The cyclic executive is essentially a table of procedure calls to be executed at the right time.

Process	Period	Computation Time
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2

Slide 10

Suppose we have an interrupt time which happens every 25ms (the minor cycle time), we can schedule the tasks as follows:

```
begin loop:
 Wait;
 A; B; C;
 Wait;
 A; B; D; E;
 Wait;
 A; B; C;
 Wait;
 A; B; D;
end loop;
```

Pros and Cons

This procedure has some advantages and disadvantages.

Slide 11

- There is no large run time system to manage, we just have a sequence of procedure calls.
- The procedures share a common address space and can easily pass data between themselves. There is no need for semaphores because concurrent access is not possible.
- All periods must a multiple of the minor cycle time.

More drawbacks

Other drawbacks include:

Slide 12

- It is difficult to incorporate sporadic processes.
- Long cycle times means a long major cycle time.
- It is difficult to construct the cyclic executive (more below).
- any process with a large computation time will need split into a fixed number of smaller units (this might go against software engineering issues).

How do I construct a cyclic Executive?

Slide 13

The actual programming of an executive is easy, but how do you actually come up with the correct sequence of events?

Constructing executives given a set of tasks is known to be an NP-complete problem.

That is to the best of our knowledge the cost of computing a schedule grows exponentially with the number of tasks.

NP completeness and scheduling

Slide 14

0The best known algorithms for scheduling essentially search for all the possible schedule upto the maximum period until one is found.

Typically heuristic schemes are used, some times find schedules (which are correct) but will sometimes not be able to find a schedule if there is one.

A typical realistic system may contain perhaps 40 minor cycles and 400 entries.

Process based Scheduling

Slide 15

There are two extremes:

- *preemptive* tasks, these are tasks that be interrupted that is preempted by another task.
- *non preemptive* tasks, these are tasks which can not interrupted but to completed once they are started.

There are various schemes in between, critical sections, deferred preemption cooperative dispatching, we will look at some of these later.

Tasks States

Slide 16

We will assume that a task is in one of two states:

- runnable;
- suspended waiting for a timing event;

When we consider sporadic tasks there is another state:

- suspended waiting for a non-timing event.

Priority

Slide 17

We shall use the following model. We have a queue of tasks to be executed. At all times the highest priority runnable task is executed.

Note that the passing of time can make a task runnable.

Rate monotonic priority assignment

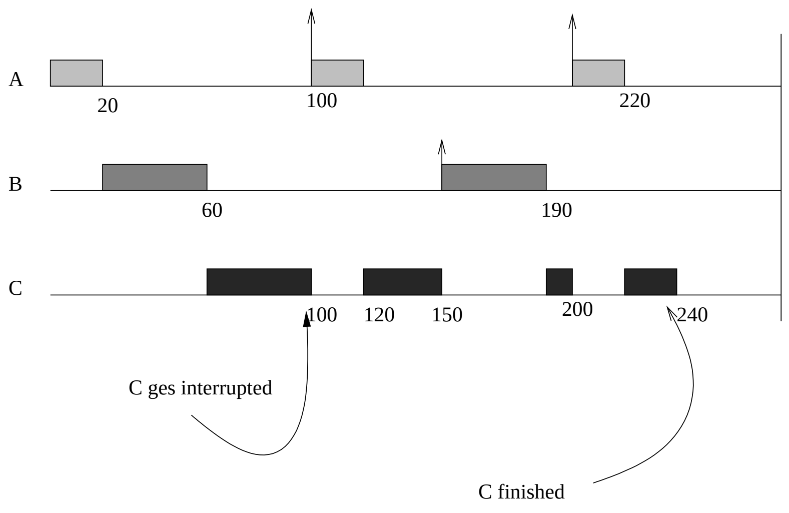
Slide 18

What if we simply assign a (unique) priority based on the tasks period where the shorter the period the higher the priority?

Consider the task set:

Process	Computation Time	Period	Priority
A	20	100	H
B	40	150	M
C	100	300	L

Slide 19



Slide 20

When does rate monotonic work?

Consider the set of tasks:

	Period	Time	Priority
A	50	12	L
B	40	10	M
C	30	10	H

This set doesn't work. Draw a picture and find out.

Utilisation

Slide 21

In 1973 Liu and Layland gave a utilisation test to see if a set of tasks can be scheduled by rate-monotonic.

The utilisation of a set of tasks is defined to be:

$$\sum_{i=1}^{N} \left(\frac{C_i}{T_i} \right)$$

Liu and Layland's theorem

Slide 22

There is theorem which states that if:

$$\sum_{i=1}^{N} \left(\frac{C_i}{T_i} \right) < N(2^{1/N} - 1)$$

then the set of tasks can be scheduled by rate monotonic.

Values of N and $N(2^{1/N} - 1)$

Slide 23

N	Bound
1	100.0
2	82.8
3	78.0
4	75.7
5	74.3
10	71.8

Limit about 69%.

For example

Slide 24

Process	Time	Period	Utilisation
A	20	100	0.2
B	40	150	0.27
C	100	350	0.28

So the total utilisation is 0.75 which is less than 0.78

Does it work both ways?

Slide 25

	Period	Time	utilisation
A	50	12	0.24
B	40	10	0.25
C	30	10	0.33

So the total utilisation is 0.82 which is greater than 0.78 *and* the task set can not be scheduled by rate-monotonic.

Completeness

Slide 26

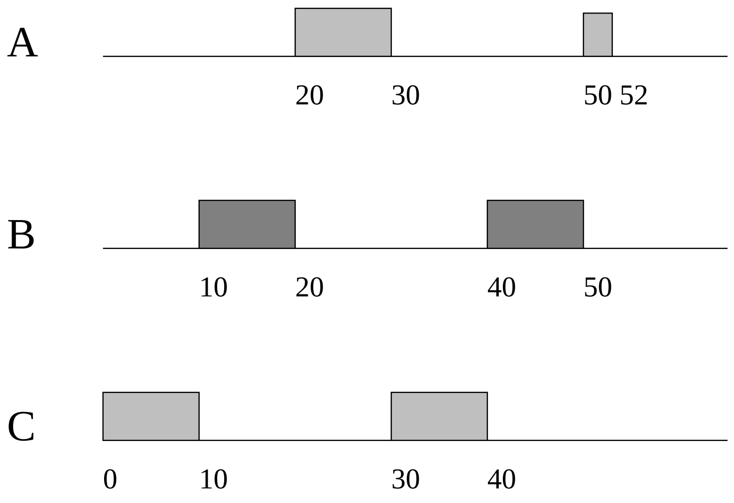
Liu and Layland's test is not exact. If a task set passes the test it can be scheduled by rate monotonic. If it fails the test it might be possible to schedule it. For example:

Name	T	D	C	Priority	Utilisation
A	52	52	12	L	0.23
B	40	40	10	M	0.25
C	30	30	10	H	0.33

Total utilisation = 0.81.

Slide 27

But we can:



Slide 28

The Next installment

- Develop an exact test.
- Extend it to more complicated situations.