So far

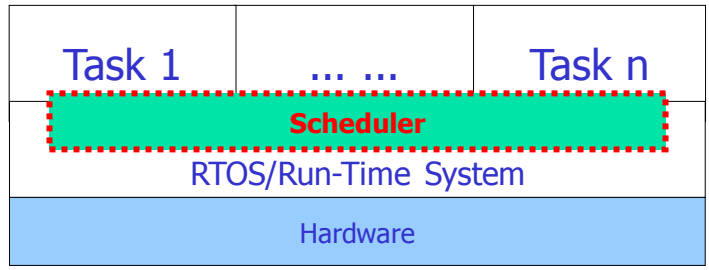
- RT programming language, Ada (as an example)
  - Real-Time Facilities (e.g. delay, delay until)
  - Concurrency/multitasking (e.g. tasking)
  - Communication/synchronization (e.g. Rendezvous)
  - Consistency in data sharing (e.g. Protected data type)
  - Run-time systems: scheduling
- RTOS
  - Basic requirements/features
    - Deterministic
    - Responsiveness
    - Users control over OS policies
    - Controlled Code size
  - Basic functions
    - OS standard functions
    - scheduling
- Synchronization & Resource sharing
  - Priority inversion problem (unbounded blocking)
  - Solutions: NPP, BIP, HLP, PCP
- WCET analysis -- Worst Case Execution Time Analysis

Today's topic

Real-Time Scheduling:

Basic concepts and basic algorithms

Overall Stucture of Real Time Systems



How to schedule the Tasks such that given timing constraints are satisfied?

3

Scheduling Problems

Given a set of tasks (ready queue)

1. Feasibility analysis: Check if all deadlines can be met using some scheduling algorithm
2. Schedulability analysis: Check if all deadlines can be met using a given scheduling algorithm

4

Task models

- Non periodic/Aperiodic Tasks
  - Appear once only
- Basic task model (3 parameters):
  - A: arrving time
  - C: computing time (resource requirement)
  - D: deadline (relative deadline)

5

Tasks with the same arrival time

Assume a list of tasks

(A, C1, D1)(A, C2, D2) ...(A, Cn,Dn)

that arrive at the same time i.e. A

- How to find a feasible schedule?

  (OBS: there may be many feasible schedules)

6

EDF [Jackson 1955]

- EDF: order tasks with nondecreasing deadlines.
- Example: (1,10)(2,3)(3,5)
  - Schedule: (2,3)(3,5)(1,10)

7

EDF: Schedulability test

- Order tasks in increasing order of deadlines
- If C1+...+Ck <=Dk for all k, the task set is schedulable
- Response time for task i, Ri =C1+...+Ci

8

EDF: Examples

- (2, 4)(1,5)(6,10) is schedulable:
  - Feasible schedule: (2,4)(1,5)(6,10)
  - Note that (1,5)(2,4)(6,10) is also feasible
- (1,10)(3,3)(2,5) is schedulable
  - The feasible schedule: (3,3)(2,5)(1,10)
  - Why not shortest task first?
- (4,6)(1,10)(3,5) is not schedulable
  - (3,5)(4,6)(1,10) is not feasible: 3+4 > 6!

9

Tasks with different arrival times (e.g. a calendar)

- Assume a list of tasks
  - S= (A1, C1, D1)(A2,C2, D2) ...(An,Cn,Dn)
- Preemptive EDF [Horn 1974]:
  - Whenever new tasks arrive, sort the ready queue according to earlist deadlines
  - Run the first task of the queue

10

Preemptive EDF: Schedulability test

- At any time, order the tasks according to EDF
  (A'_1, C'_1, D'_1) … … (A'_i, C'_i, D'_i)
- If $C'_1+...+C'_k <= D'_k$ for all k=1,2...i, then the task set is schedulable at the moment
- If S is schedulable at all time points at which tasks arrive, S is schedulable

11

Preemptive EDF: Example

Consider (1, 5, 11)(2,1,3)(3, 4,8)

- Deadlines are relative to arrival times
- At 1, (5,11)
- At 2, (1,3)(4,10)
- At 3, (4,8)(4,9)

12

Optimality of scheduling algorithms

- Algorithm A is optimal in a class of algorithms if any algorithm in the class can find a feasible schedule for a task set (i.e. meet all the deadlines), A can also do it

- An algorithm can be optimal in the sense of optimizing the system behaviors e.g. Minimizing the maximum latency, response time, throughput etc

13

Preemptive EDF: Optimality

- Assume that Ri is the finishing time/response time of task i. Let Li = Ri-Di (the lateness for task i)

- FACT: preemptive EDF is optimal in minimizing the maximum lateness L_{max}= MAXi(Li)
  - Note a task set is schedulable if L_{max} <= 0
  - This implies:

EDF optimality: if any algorithm can schedule a task set and meet all the deadline, EDF can do it too.

14

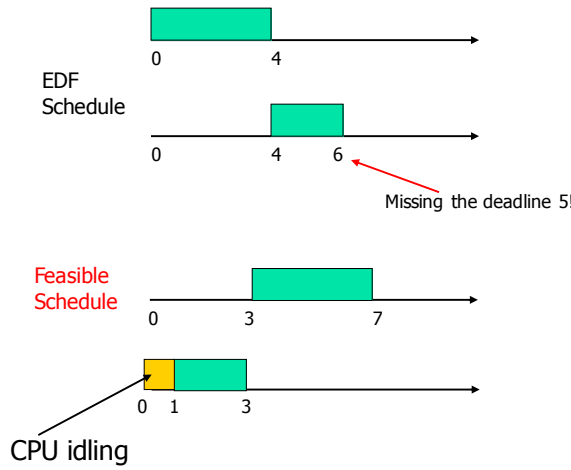
Non preemptive EDF

- Run a task until it's finished and then sort the queue according to EDF
  - \+ Easy to implement, less overhead (no more context switch than necessay)
  - \- However it is not optimal, it may not find the feasible schedule even it exists.

15

Non-preemptive EDF: example

	T1	T2
A	0	1
C	4	2
D	7	5



16

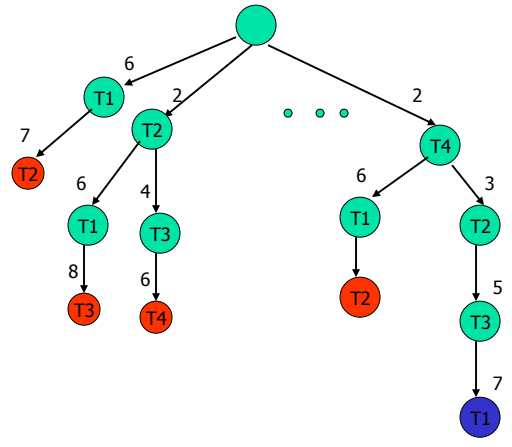
To construct ”optimal” non-preemptive Schedule: ”complete search, NP-hard” is needed

- The decision should be made according to all the parameters in the whole list of tasks
- The worst case is to test all possible combinations of n tasks (NP-hard, difficult for large n)

17

Example (Bratley’s alg.)

	T1	T2	T3	T4
A	4	1	1	0
C	2	1	2	2
D	7	5	6	4



18

Heuristic methods

- Similar to Bratley's alg. But
  - Use heuristic function H to guide the search until a feasible schedule is found, otherwise backtrack: add a new node in the search tree if the node has smallest value according to H e.g H(task i) = Ci, Ai, Di etc [Spring alg.]
  - However it may be difficult to find the right H

19

Example Heuristics

- H(Ti) = Ai FIFO
- H(Ti) = Ci SJF
- H(Ti) = Di EDF
- H(Ti) = Di +w*Ci EDF+SJF
- ...

20

EDF: + and –

- Preemptive EDF
  - Simple (+)
  - Optimal (+)
  - No need for computing times (+)
  - ”difficult” to implement efficiently (-)
    - Need a list of ”timers”, one per task,
    - Overheads for context switch
- Non-preemptive EDF
  - Minimal context switch (+)
  - It is not optimal (-)

21

Other scheduling algorithms

- Classical ones
  - HPF (priorities = degrees of importance of tasks)
  - Weighted Round Robin
- LRT (Latest Release Time or reverse EDF)
- LST (Least Slack Time first)

22

Latest Release Time (reversed EDF)

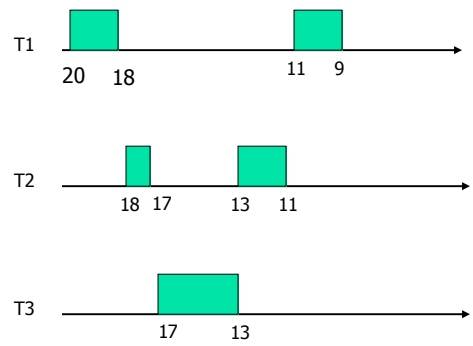
- Release time = arrival time
- Idea: no advantage to completing any hard task sooner than necessary. We may want to postpone the execution of hard tasks e.g to improve response times for soft tasks.
- LRT: Schedule tasks from the latest deadline to earliest deadline. The latest 'arrival time' first.
  - If any task can not be completed before its "arrival time", it is non-schedulable by LRT
- FACT: LRT is optimal in finding feasible schedule (for preemptive tasks)

23

LRT: Example

	T1	T2	T3
A	0	11	12
C	4	3	4
D	20	18	17

(D=absolute deadline)



Reverse time: we get the schedule:
T1(9,11)T2(11,13)T3(13,17)T2(17,18)T1(18,20)
OBS: from 0 to 9, soft tasks may be running!

24

LRT: + and -

- It needs Arrival times (-)
- It got to be an off-line algorithm (-)
- Only for preemptive tasks (-)
- It could optimize Response times for soft tasks (+)

25

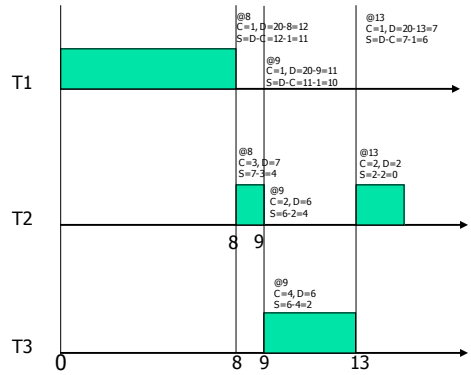
Least slack time first (LST)

- Let Si = Di-Ci (the Slack time for task i)
  - Si is the maximal (tolerable) time that task i can be delayed
- Idea: there is no point to complete a task earlier than its deadline. Other (soft) tasks may be executed first
  - Slack stealing
- LST: order the queue with nondecreasing slack times
- FACT: preemptive LST is optimal in finding feasible schedules

26

LST: Example

	T1	T2	T3
A	0	8	9
C	9	3	4
D	20	7	6



Comment: a task should run until a Slack reaches 0 (to avoid context switch)
And if more than one 0-slack: nonschedulable

LST: + and –

- It needs Computing times (-)
- Only for preemptive tasks (-)
- Not easy to implement! (-)
- But it can run on-line (+)
- and it may improve response times?

Summary: scheduling independent tasks

Task types	Same arrival times	Preepmtive Different arrival times	Non preemptive Different arrival times
Algorithms For Independent tasks	EDF,Jackson55 O(n log n), optimal	EDF, Horn 74 O(n**2), Optimal LST, LRT optimal	Tree search Bratley'71 O(n*n!), optimal Spring, Stankovic et al 87 $O(n^2)$, Heuristic

29

Dependent tasks

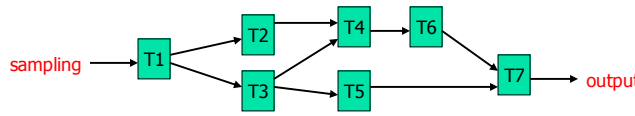
- We often have conditions or constraints on tasks e.g.
  - A must be computed before B
  - B must be computed before C and D
- Such conditions are called precedence constraints which can be represented as *Directed Acyclic Graphs* (DAG) known as Precedence graphs

- Such graphs are also known as **"Task Graph"**

30

Dependent tasks: Examples

- Input/output relation (conjunctive/disjunctive incomming edges)
  - Some task is waiting for output of the others, data flow diagrams

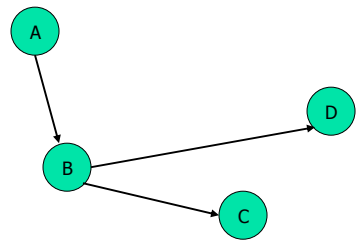


  - Individual and End-to-end deadlines

31

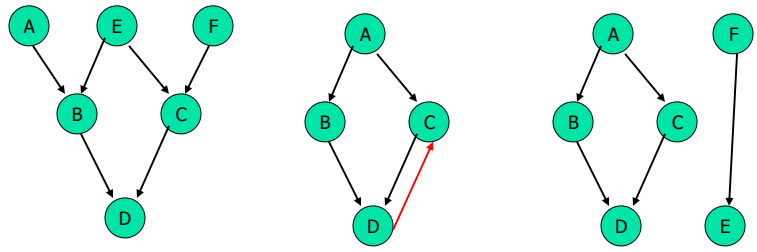
Precedence graph: Example

- A must be computed before B
- B must be computed before C and D



32

Precedence graph: Examples



Not a precedence graph!

We consider only conjunct join!

Dependent tasks with the same arrival times

- Assume a list of tasks:
  (A,C1,D1)(A,C2,D2) ...(A,Cn,Dn)
- In addition to the deadlines D1...Dn, the tasks are also constrained by a DAG
- Solution: Latest Deadline First (LDF), Lawler 1973
- FACT: LDF is optimal (in finding feasible schedules)

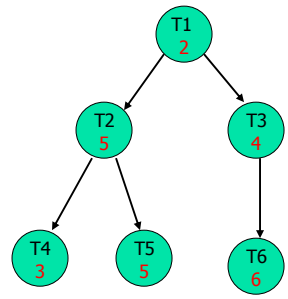
Latest Deadline First (LDF)

- It constructs a schedule from tail to head using a queue:
  1. Pick up a task from the current DAG, that
     - Has the latest deadline and
     - Does not precede any other tasks (a leaf!)
  2. Remove the selected task from the DAG and put it to the queue
- Repeat the two steps until the DAG contains no more tasks. Then the queue is a potentilly feasible schedule. The last task selected should be run first.
- Note that this is similar to LRT

35

LDF: Example

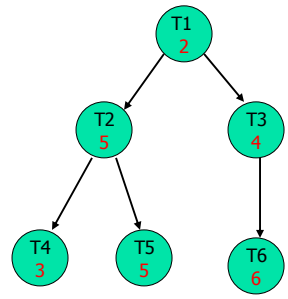
	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6



36

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

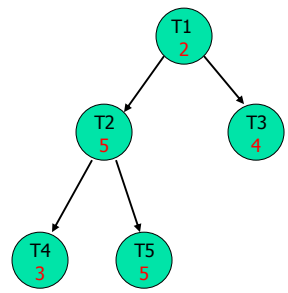


LDF: T6

37

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

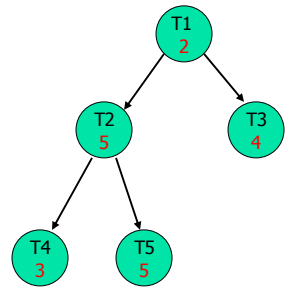


LDF: T6

38

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

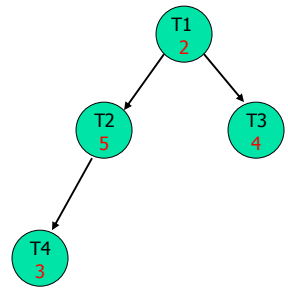


LDF: T6,T5

39

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

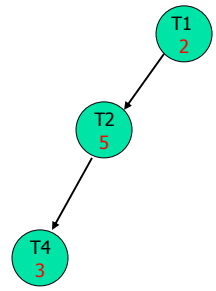


LDF: T6,T5

40

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

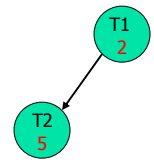


LDF: T6,T5,T3

41

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6



LDF: T6,T5,T3,T4

42

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

LDF: T6,T5,T3,T4,T2

43

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

LDF: T6,T5,T3,T4,T2,T1

44

LDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

LDF: T6,T5,T3,T4,T2,T1

Feasible Schedule

45

Earlest Deadline First (EDF)

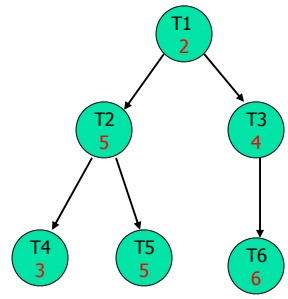
- It is a variant of LDF, but start with the root of the DAG:
  1. Pick up a task with earlest deadline among all nodes that have no fathers (the roots)
  2. Remove the selected task from the DAG and put it to the queue
- Repeat the two steps until the DAG contains no more tasks. Then the queue is a feasible schedule.

- Unfortunately, EDF is not optimal
  - The "earliest deadline now" may lead to "longer deadline in future"
  - (see the following example)

46

EDF: Example

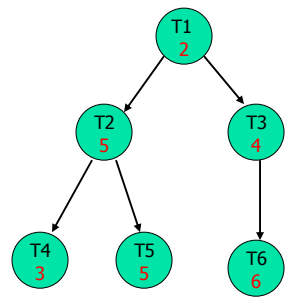
	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6



47

EDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

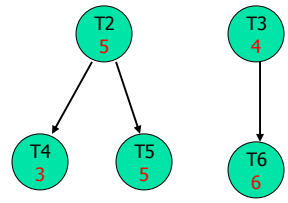


EDF: T1

48

EDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

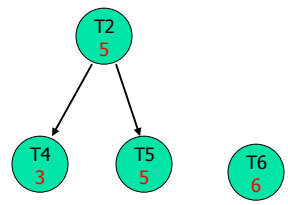


EDF: T1

49

EDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6



EDF: T1,T3

50

EDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

EDF: T1,T3,T2

51

EDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

EDF: T1,T3,T2,T4,T5,T6

52

EDF: Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6

T4 will miss its Deadline: 3

LDF: T6,T5,T3,T4,T2,T1
Feasible

EDF: T1,T3,T2,T4,T5,T6
Infeasible

53

Dependent tasks with different arrival times

- Assume a list of tasks:
  S = (A1,C1,D1)(A2,C2,D2)...(A3,Cn,Dn)
- In addition to the deadlines D1...Dn, the tasks are also constrained by a DAG
- Solution: Complete Search
  - The Bratley's algorithm (search and backtrack whenever needed)

54

Better algorithms?

- Assume a list of tasks:

  S = (A1,C1,D1)(A2,C2,D2)...(A3,Cn,Dn)
- In addition to the deadlines D1...Dn, the tasks are also constrained by the task graph
- Idea:
  - Transform the task set S to an Independent task set S* such that S is schedulable under DAG iff S* is schedulable

55

Idea: how to transform S to S*?

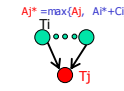
- Idea:

  If Ti ->Tj is in the DAG i.e. Ti must be executed before Tj, we replace the arrival time for Tj and deadline for Ti with
  - Aj* = max(Aj, Ai+Ci)
    - Tj can not be computed before the completion of Ti
  - Di*=min(Di,Dj-Cj)
    - Ti should be finished early enough to meet the deadline for Tj

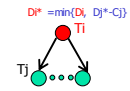
56

Algorithm (EDF*): transform S to S*

- Let arrival times and deadlines be 'absolute times'
- Step 1: Transform the arrival times from roots to leafs



  - For all initial (root) nodes Ti, let Ai* = Ai
  - REPEAT:
    - Pick up a node Tj whose fathers' arrival times have been modified. If no such node, stop. Otherwise:
    - Let Aj* =max(Aj, max{Ai*+Ci: Ti->Tj})
- Step 2: Transform the deadlines from leafs to roots



  - For all terminal (leafs) nodes Tj, let Dj* = Dj
  - REPEAT:
    - Pick up a node Ti all whose sons deadlines have been modified. If no such node, stop. Otherwise:
    - Let Di* =min(Di, min{Dj*-Cj: Ti->Tj})
- Step 3: use EDF to schedule S*=(A1*,C1,D1*)...(An*.Cn,Dn*)

57

EDF*: optimality

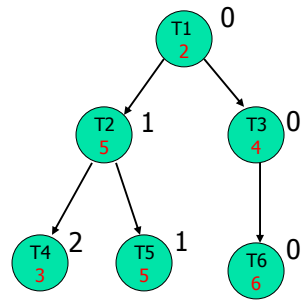
FACT:

- S is schedulable under a DAG iff S* is schedulable
- EDF* is optimal in finding a feasible schedule

58

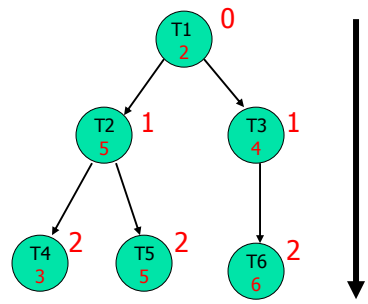
Example

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6
A	0	1	0	2	1	0



EDF*: Example(1)

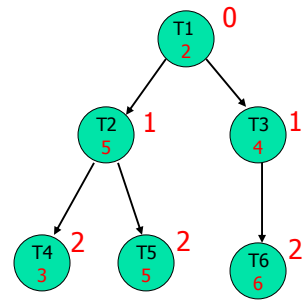
	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6
A	0	1	0	2	1	0



Step 1: Modifying the arrival times (top-down)

EDF*: Example(1)

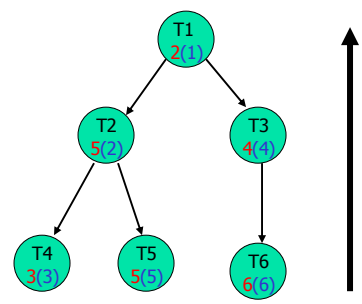
	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6
A*	0	1	1	2	2	2



Step 1: Modifying the arrival times (top-down)

EDF*: Example(2)

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6
A*	0	1	1	2	2	2

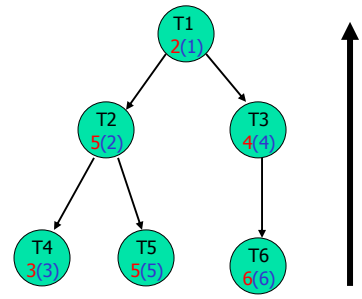


Step 2: Modifying the deadlines (bottom-up)

EDF*: Example(2)

S*

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D*	1	2	4	3	5	6
A*	0	1	1	2	2	2



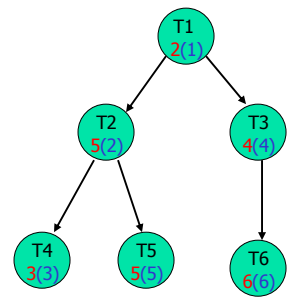
Step 2: Modifying the deadlines (bottom-up)

63

EDF*: Example(3)

S*

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D*	1	2	4	3	5	6
A*	0	1	1	2	2	2



Step 3: now we don't need the DAG any more!

64

EDF*: Example(3)

S*

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D*	1	2	4	3	5	6
A*	0	1	1	2	2	2

Step 3: schedule S* using EDF

EDF*: Example(3)

S*

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D*	1	2	4	3	5	6
A*	0	1	1	2	2	2

Finally we have a schedule: T1,T2,T4,T3,T5,T6

Summary: scheduling aperiodic tasks

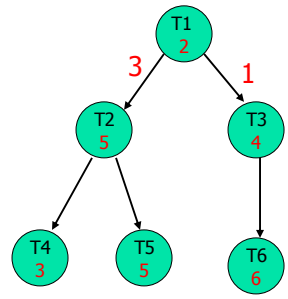
Task types	Same arrival times	Preepmtive Different arrival times	Non preemptive Different arrival times
Algorithms for Independent tasks	EDF,Jackson55 O(n log n), optimal	EDF, Horn 74 O(n**2), Optimal LST, optimal LRT, optimal	Tree search Bratley'71 O(n n!), optimal Spring, Stankovic et al 87 O(n**2) Heuristic
Algorithms for Dependent tasks	LDF, Lawler 73 O(n**2) Optimal	EDF* Chetto et al 90 O(n**2) optimal	As above

More about "TASK GRAPH"

- Acyclic graphs or Trees where
  - Nodes are "computation tasks"
  - Edges are "communication links"
- Constraints on task graphs
  - Each node is characterized by "resource requirements" and "local deadline"
  - Each edge may have a "communication delay"
  - End-to-End deadline: root to leaf
- How to implement a "Task graph"?
  - Single processor systems (communication delay = 0)
  - Multiprocessor systems?
    - Communication bus e.g. CAN BUS
    - Shared memory, memory hiearchy
    - Network-on-Chip

Example: task graph with communication delays

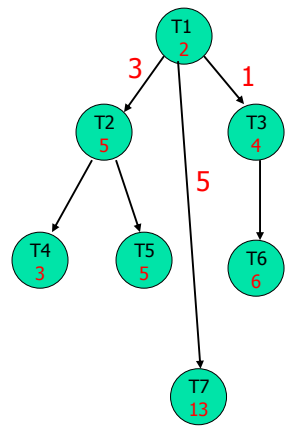
	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6



69

Example: task graph with communication delays

	T1	T2	T3	T4	T5	T6
C	1	1	1	1	1	1
D	2	5	4	3	5	6



You may have to schedule
the communication bandwidth"

70