Real Time Systems 2

Assignment With Precedence Conditions Algorithm

Lecture 9

Assignment With Precedence Conditions

- Algorithm that assigns and schedules tasks with precedence conditions and additional resource constraints.

- The basic algorithm idea is to reduce communication costs by assigning (if possible) to the same processor tasks that heavily communicate with one another.

Assignment With Precedence Conditions

- Each task may be composed of one or more subtasks. the release time of each task and the worst-case execution time of each subtask are given.
- The subtask communication pattern is represented by a task precedence graph that implicitly defines the precedence condition.

- The communication cost between tasks must determined, so if subtask s1 sends output to s2, this is done at the end of s1 execution and must arrive before s2 can being executing.

Assignment With Precedence Conditions

- Associated with each subtask is a latest finishing time (LFT) that will be explained through an example;
- The algorithm is a trial-and-error process. Assign set of subtasks to the processors one by one in the order of their LFT values.
- If two subtasks have the same LFT value, so the subtask with the greatest number of successor it wins.
- Check for feasibility after each assignment, if one assignment is not feasible try another one, etc.

Assignment With Precedence Conditions

- If no assignment works for this particular subtask, we backtrack and try changing the assignment of the previously subtask and continue from there.
- Subtasks that communicate with each other a lot are assigned to the same processor if this allows a feasible scheduling.
- A threshold policy ***kc*** is followed for this. If ei and ej are the execution time times of subtasks si and sj and cij is the volume of communication between them, we try to assigning si and sj to the same processor if: **$e_i + e_j / c_{ij} < K_c$**

Assignment With Precedence Conditions

- The idea of ***kc*** is to balance the benefits of assigning subtasks to the same processor (the communication cost is zero) against the potential benefits of assigning them to different processors.
- The assignment is done in part by checking each pair of communicating subtasks to see if they must be assigned to the same processor.
- May be some subtask be on same processor even though there is no communication between them.

Assignment With Precedence Conditions

- kc is a tunabel parameter, if ***kc*** is 0, then inter-subtask communication will not be taken in consideration when we do the assignment.
- If ***kc*** is too high, then most of the time, we will be forced to assign tasks to the same processor, and this may result in an infeasible schedule.
- When we obtain an infeasible schedule, we are forced to reduce kc adaptively to relax this condition.

Assignment With Precedence Conditions

- The algorithm can be described as follows:
- 1. Start with the set of tasks that need to be assigned and scheduled. Choose a value for ***kc*** and determine based on this value, which tasks need to be on the same processor.
- 2. Start assigning and scheduling the subtasks in order of precedence. If a particular allocation is infeasible, reallocate if possible.
- If feasibility is impossible to achieve because of the tasks that need to be assigned to the same processor, reduce ***kc*** suitably and go to the previous step.

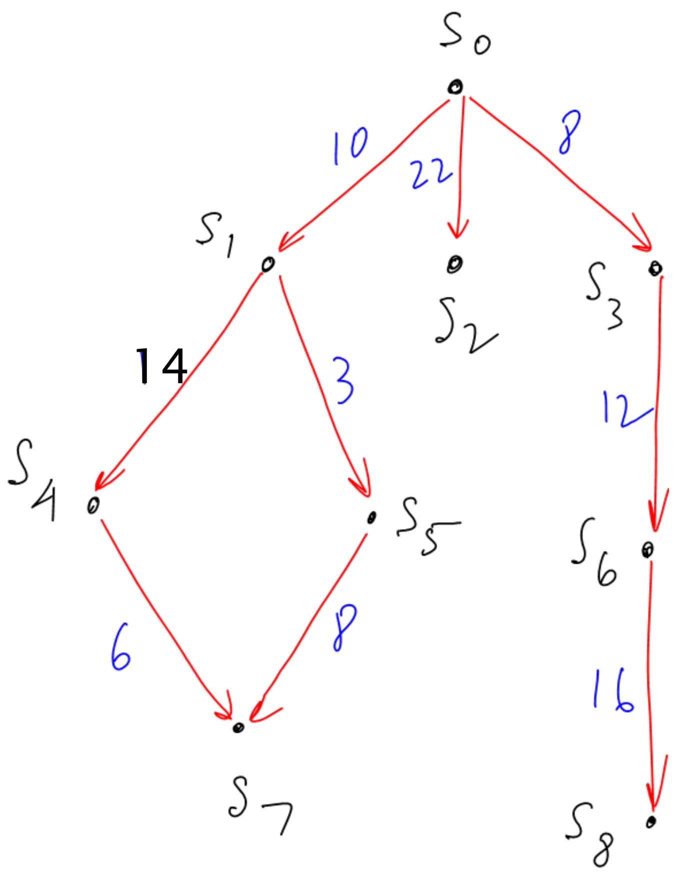
Assignment With Precedence Conditions

- 3. Stop when either the tasks have been successfully assigned and scheduled, or when the algorithm has tried more than a certain number of iterations,
- In the former case, output the completed schedule, in the latter, declare failure.

Example: Task assignment and scheduling algorithm

- Consider a system composed of one task with 8 subtasks. The table below gives the execution times and the deadline for the output to the outside world.
- The labels within the circles are the subtasks and the arc labels denote the volume of communication cost between tasks.
- The task graph indicates the precedence and the communication volume.
- Apply LFT and come up with a feasible schedule. Note that the value of ***kc*** can be relaxed in order to obtain a feasible constraint. Calculate the LFT values for all subtasks. For generating a feasible schedule start with ***kc*** = 1.5

Example: Task assignment and scheduling algorithm



Subtask	ei	Di	LFT	pair	ei+ej/cij
S0	4	–	22–15=7 Or 24–10=14 Or 26–4=22	S0,S1=10	4+10/10=1.4
S1	10	–	42–18=24 Or 42–3=39	S0,S2=22	4+15/22=0.86
S2	15	22	22	S0,S3=8	4+4/8=1
S3	4	–	32–6=26	S1,S4=14	10+18/14=2
S4	18	–	45–3 = 42	S1,S5=3	10+3/3= 4.33
S5	3	–	45–3 = 42	S3,S6=12	4+6/12=0.83
S6	6	–	40– 8 = 32	S4,S7=6	18+3/6=3.5
S7	3	45	45	S5,S7=8	3+3/8=0.75
S8	8	40	40	S6,S8=16	6+8/16=0.88

- We have 1 task with 8 subtasks

LFT S0= d2– e2 =22 - 15=7
LFT S1= d7– e7– Max{ e4,e5}= 45 -3 -18=24
LFT S2=22, S7=45, S8=40 “outside world”

Example: Task assignment and scheduling algorithm

- Solution: the deadline of the "outside world" is S2=22, S7=45, S8=40.
- If we set ***kc*** =1.5, so the pairs are assigned to same processor: {S0,S1}, {S0,S2}, {S0,S3}, {S3,S6}, {S5,S7}, {S6,S8},
- These implies that tasks must all assigned to the same processor: S0,S1,S2,S3,S6,S8.
- Suppose we have a system consisting of two processors P0,P1 on which we seek to assign these tasks.
- Single bus used for intercommunication network.

Example: Task assignment and scheduling algorithm

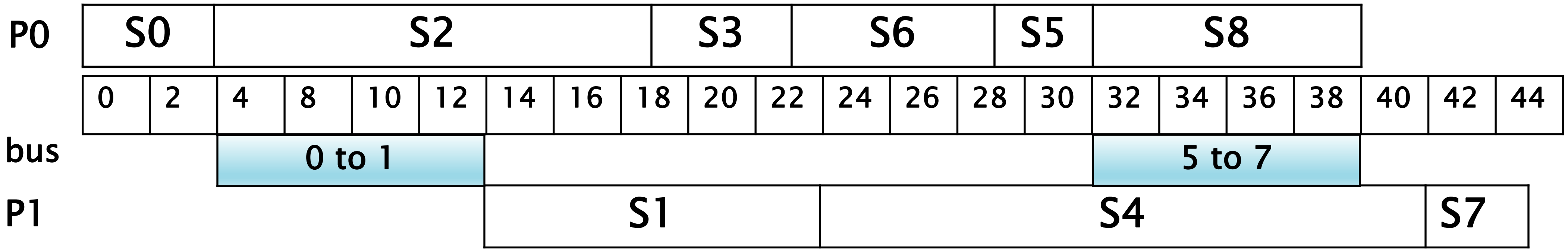
- 1- Assign S0 to P0 at interval [0,4], assume time released is 0.
- 2- Assign S1 to P0 at interval [4,14], no communication are incurred , all on same processor.
- 3- Assign S2 to P0 its clear we cannot schedule S1 and S2 both on same processor (e1=10,e2=15) so that they complete before their LFT.
- Therefor we must reduce ***kc*** to 1 ,so the pair here are: {S0,S2}, {S3,S6}, {S5,S7}, {S6,S8}.
- 4- Now assign S1to P1 over [14,24], also schedule the network path between P0 and P1 for [4,14], the communication between S0 and S1.
- 5- Assign S2 to P0 at interval [4,19].
- 6- Assign S3 to P0 at interval [19,23].

Example: Task assignment and scheduling algorithm

- 7- Assign S4 to P0 at interval [23,41].
- 8- Assign S5 to P0 at interval [4,19], this is infeasible , so backtrack to the previous step.
- 9- Reassign S4 to P1 at interval [24,42].
- 10- Reassign S6 to P0 at interval [23,29].
- 11- Assign S5 to P0 at interval [29,32].
- 12- Assign S7 to P0 [32,35] is infeasible because after that S8 will end to 43, [35,43], so assign S7 to P1at interval [42,45].
- 13- Assign S8 to P0 at interval [32,40].
- If the task set is periodic , then we only need to consider the tasks released during the least common multiple of the task periods.

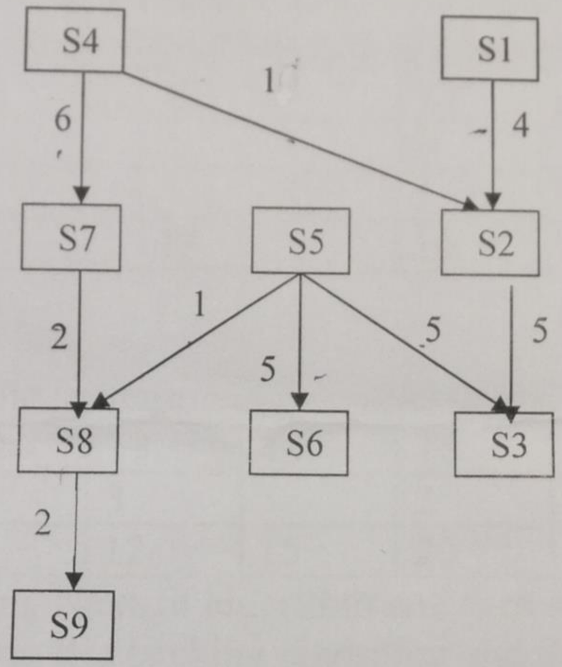
Example: Task assignment and scheduling algorithm

- If ei +ej/cij < **Kc** we try to assigning si and sj to the same processor.



HW:

Consider the following precedence graph:



And

Subtask	S1	S2	S3	S4	S5	S6	S7	S8	S9
exi	2	2	2	4	3	3	5	1	1
Di			17			10			22

a. Calculate the latest finishing time of every subtask.
b. Allocate and schedule them using 2 CPUs and Kc=1.5.