Scheduling Algorithms

Contents

Cyclic Executives

…

Review: Realtime task representations

- A periodic task Ti can be represented by a 4 tuple: $(\emptyset_i, P_i, e_i, D_i)$
- If using 3 tuple (P_i, e_i, D_i), it is equivalent to $(0, P_i, e_i, D_i)$
- If using 2 tuple (P_i, e_i), it is equivalent to $(0, P_i, e_i, P_i)$

Cyclic Executive (CE)

- It gives off-line table-driven static-schedule which specifies exactly when each job executes
  - Asssumptions:
    - Parameters of jobs with hard deadlines known
    - Task scheduling is non-preemptive
  - Non-periodic work can be run during time slots not used by periodic tasks
  - Sophisticated algorithms can be used

- Consider 4 periodic tasks

  T1 = (4; 1),

  T2 = (5; 1.8),

  T3 = (20; 1),

  T4 = (20; 2)

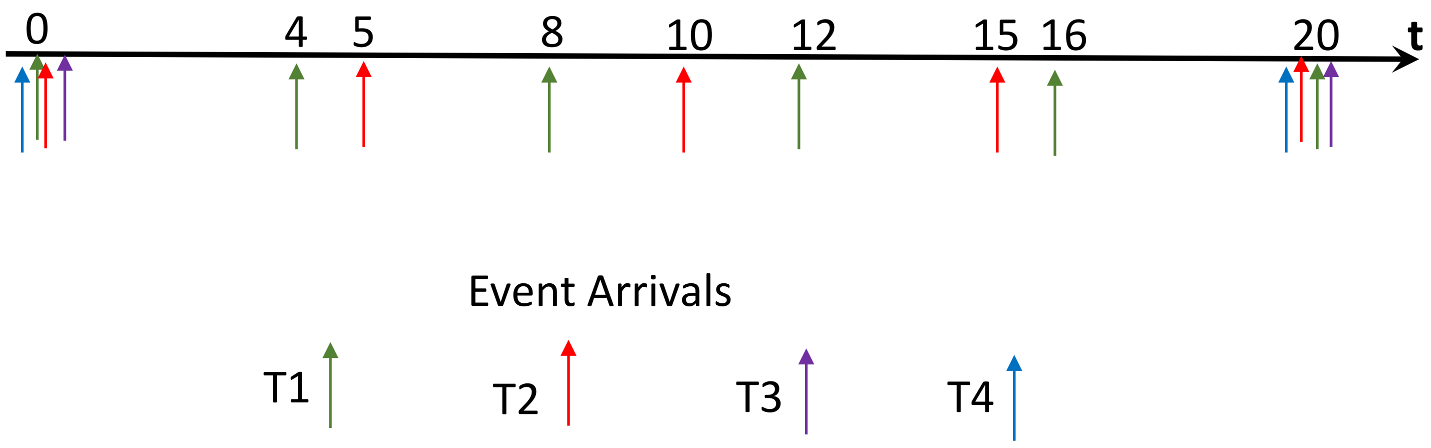
Questions

What is total utilization?

How to construct schedule for these processes?

A Possible Schedule

Tasks:
T1 = (4; 1), T2 = (5; 1.8), T3 = (20; 1), T4 = (20; 2)



Time	Process
0	T1
1	T2
2.8	T3
3.8	T4
5.8	T1
6.8	T2
8.6	T1
9.6	Idle

....

Hyperperiod

- Hyperperiod is defined as the least common multiple (lcm) of the periods of all the periodic tasks.

- The (maximum) number of arriving jobs in a hyperperiod is denote as N, so $\mathrm{N} = \sum_{i=1}^{n} H/p_i$, where p_i is period of task i.

- In the given example, hyperperiod H is 20 for the four tasks, and $\mathrm{N} = 11$.

Tasks:
T1 = (4; 1)
T2 = (5; 1.8)
T3 = (20; 1)
T4 = (20; 2)

Frames

- We wish that scheduling decisions made at regular intervals rather than at arbitrary times.

- We divide a hyperperiod into frames
  - Timing is enforced only at frame boundaries
  - Each task must fit within a single frame
  - Multiple tasks may be executed in a frame
  - Frame size is f
  - Number of frames per hyperperiod is $F = H/f$

Frame Size Constraints

C1: A job/instance must fit into a frame. So $\mathrm{f} \geq \max_{1 \leq i \leq n} e_i$ for all tasks.

Justification: Non-preemptive tasks should finish executing within a single frame

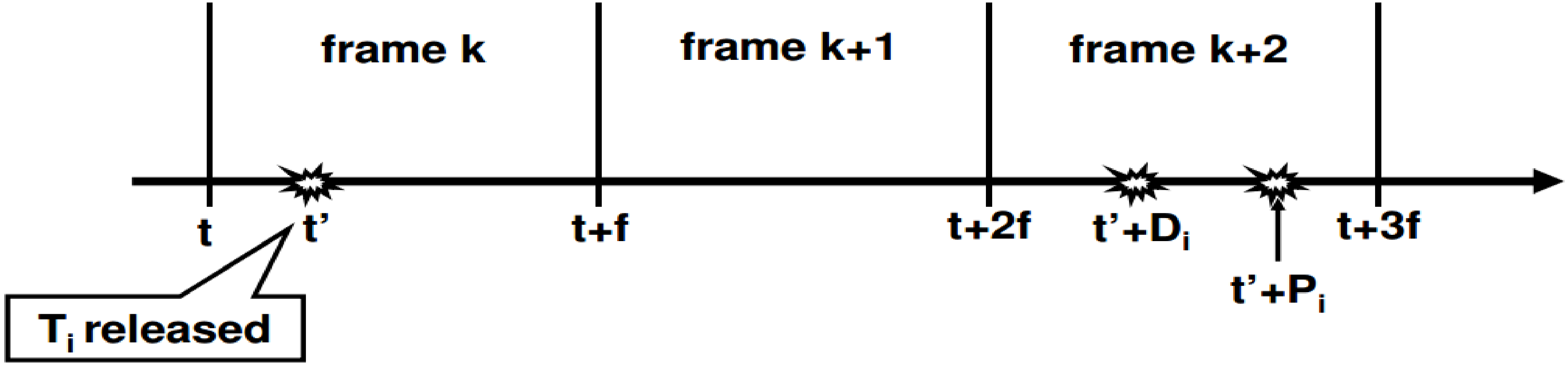
C2: H must be evenly divided by f, i.e., the hyperperiod has an integer number of frames.

Justification: Keep the cyclic schedule table size small

Frame Size Constraints

C3: f should be sufficiently small, so that there should be a complete frame between the release and deadline of every task

Justification: Schedule the task before deadline missing



Therefore: **2f – gcd (Pi, f) ≤ Di** for each task **i**

More Explanation

- Refer to figure in previous page. t denotes the beginning kth frame, and task Ti is released at t'. In order to have one full frame between release time and deadline of the job, we have:

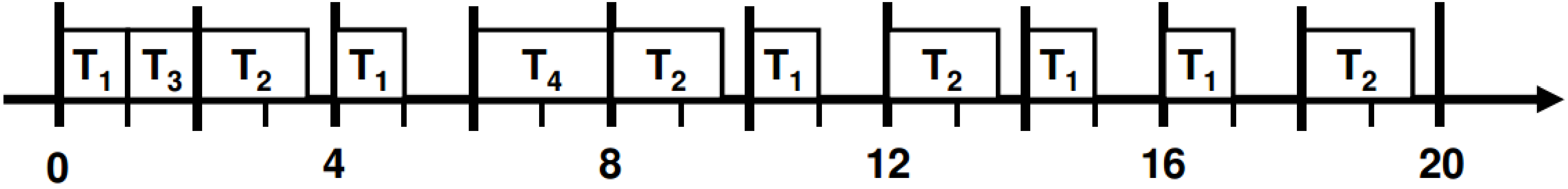
$$2f - (t' - t) \leq D_i$$

- The difference $t' - t$, is at least $\mathrm{gcd}(P_i, f)$, which result in the following constraint

$$2f - gcd(Pi, f) \leq D_i$$

Example Revisited

- Consider a system with four tasks: T1 = (4,1), T2 = (5, 1.8), T3 = (20, 1), T4 = (20, 2), and H = lcm (4,5,20) = 20
- What value should f take?
  - By Constraint 1: f ≥ 2
  - By Constraint 2: f might be 1, 2, 4, 5, 10, or 20
  - By Constraint 3: only f=2 works



In Class practice

- Given the task set:T1(6; 1); T2(10; 2); T3(18; 2)

1. Determine the size of the hyperperiod.
2. Find all suitable frame size(s)
3. Using frame size 6, construct off-line schedule for CE implementation

Solution

1. LCM(6,10,18)=90
2. C1: f>=2

   C2: 2, 3, 5, 6, 9,10,15,18…

   C3: 2f-gcd(pi,f)<=Di

   f=2, T1: 4-2<6, T2: 4-2<10, T3: 4-2<18 (OK)

   f=3, T1: 6-3<6, T2: 6-1<10, T3: 6-3<18 (OK)

   f=5, T1: 10-1>6 (X)

   f=6, T1: 12-6=6, T2: 12-2<10, T3: 12-6<18 (OK)

   f=9, T1: 18-3>6

   f=10, T1:20-2>6

   f=15, T1:30-3>6

   f=18, T1:36-2>6

3. When f=6, every frame has a slot for T1.

   f1: T1, T2, T2, T3, T3, I

   f2: T1, I, I, I, I, I

   f3: T1, T2, T2, I, I, I

   f4: T1, T3, T3, I, I, I

   f5: T1, T2, T2, I, I, I

   f6: T1, T2, T2, I, I, I

   f7: T1, T3, T3, I, I, I

   …

Task Slices

- What if frame size constraints cannot be met?
  - Example: T = { (4, 1), (5, 2, 7), (20, 5) }
  - By Constraint 1: f ≥ 5
  - By Constraint 3: f ≤ 4

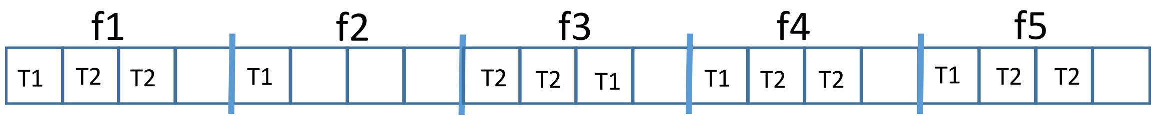
- Solution: “slice” a task into smaller sub-tasks, so T3=(20, 5) becomes T31=(20, 1), T32=(20, 3), and T33= (20, 1). Now f = 4 works

- Question: Why not split T_3 into T_{31}= (20, 2) and T_{32} = (20, 3)?

Answer:

- *T1 with a period of 4 must be scheduled in each frame of size 4*
- *T2 with a period of 5 must be scheduled in 4 out of 5 frames*
- *This leaves only 1 frame with 3 units of time for T3, other frames have only 1 unit of time and cannot have a job with execution time of 2.*

Hyperperiod = 20



Hence, we have to split 5 to 1, 3, 1.

Note that splitting tasks is painful and error prone!

Cyclic Scheduling Decision Summary

- Three decisions:

  Step 1. Choose frame size (Consider the 3 constraints)

  Step 2. If a suitable frame size is found, go to Step 3;

  otherwise, break constraint 1 to select a smaller frame size satisfying C2 & C3, and partition tasks into slices.

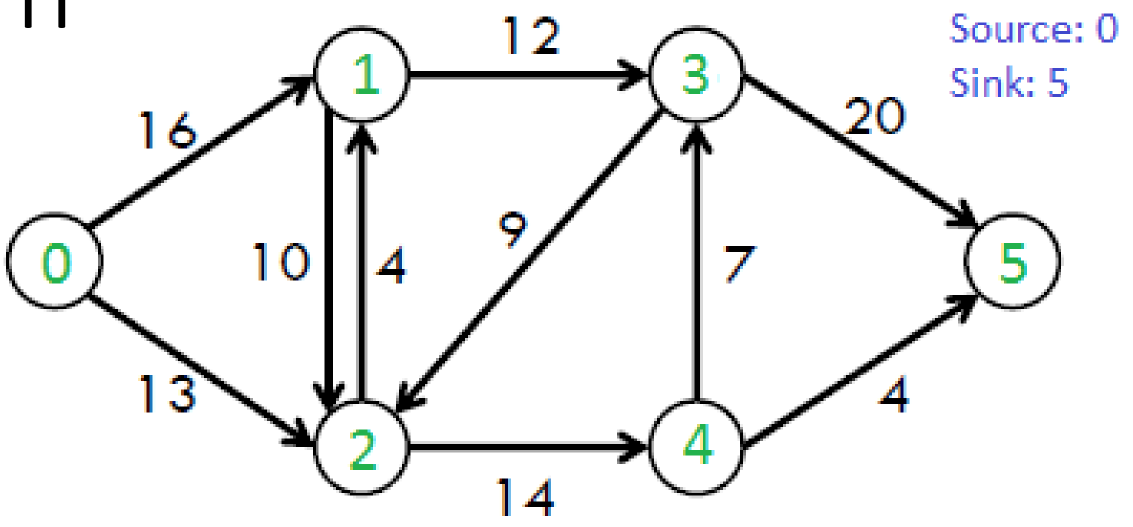
  Step 3. Place jobs/slices into frames

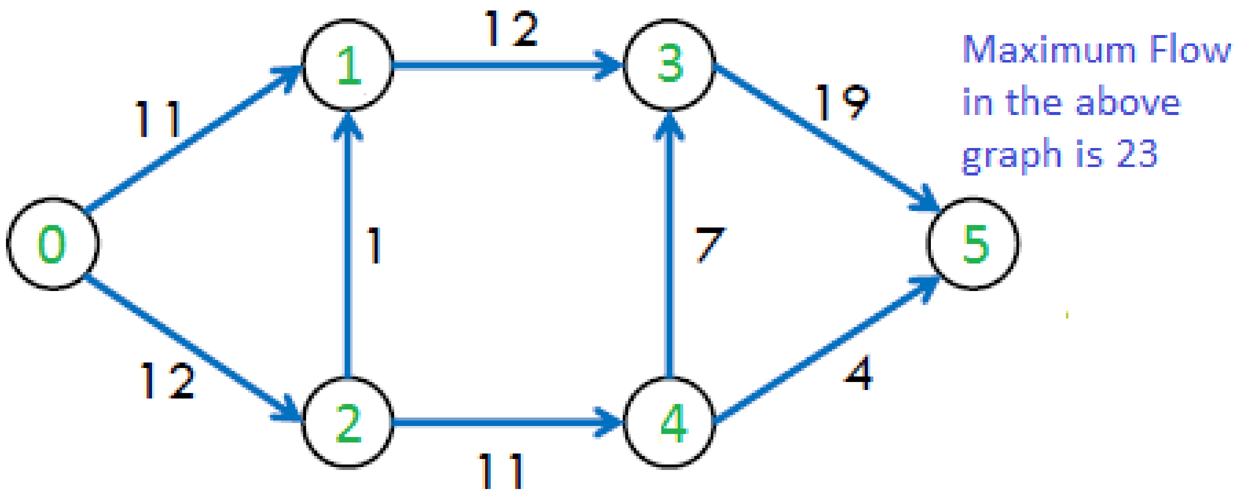
- In general these decisions are not independent.

- Try to partition the job into as few slices as necessary to meet frame size constraints but if there is no feasible schedule, split the job into smaller slices.

Network Flow Problem

- Given a graph of links, each with a fixed capacity, determine the maximum flow through the network
- Efficient algorithms exist to get the maximum flow



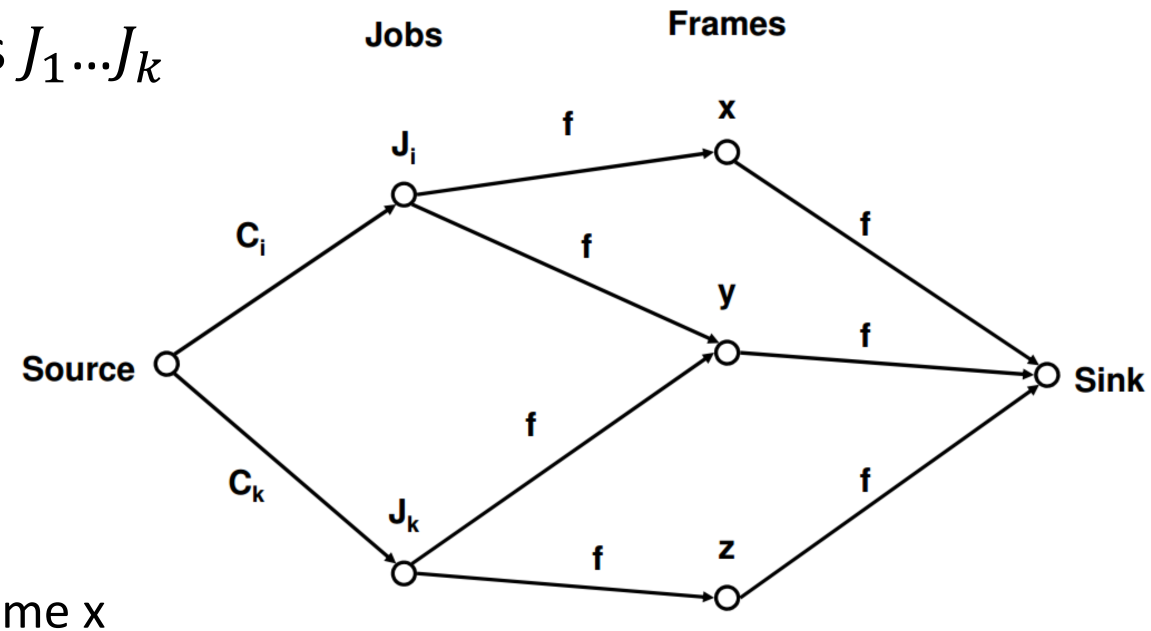


How to Derive Frame Size And Schedule Meeting All Constraints?

- Solution: Reduce to a network flow problem
  - Use constraints to compute all possible frame sizes
  - For each possible size, try to find a schedule using network flow algorithm
    - If max flow has the same value as the total execution time of all instances:
      - A schedule is found and we're done
    - Otherwise:
      - Schedule is not found, look at the next frame size
    - If no frame size works, system is not schedulable using cyclic executive

Flow Graph

- Denote all jobs in hyperperiod of F frames as $J_1 \ldots J_k$
- Vertices:
  - k job vertices $J_1, J_2, \ldots J_k$
  - F frame vertices x, y, …,z



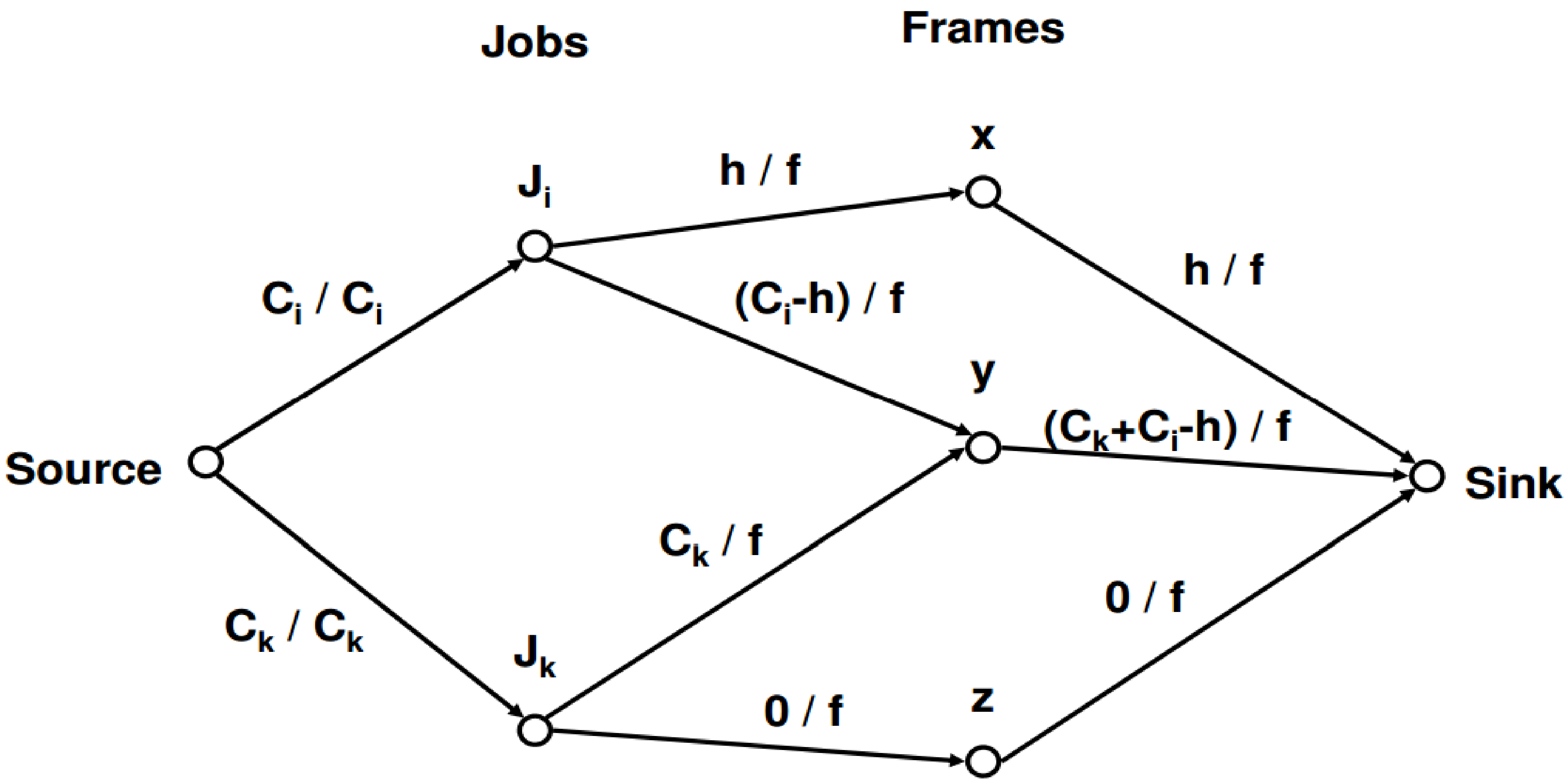
- Edges:
  - (source, Ji) with capacity Ci (=ei)
    - Encodes jobs' compute requirements
  - (Ji, x) with capacity f, iff Ji can be scheduled in frame x
    - Encodes periods and deadlines
    - An edge is connected between a job node and a frame node if two the following 2 conditions are met: (1) the job arrives **before** or at the starting time of the frame; (2) the job's absolute deadline is **larger than** or equal to the ending time of the frame
  - (f, sink) with capacity f
    - Encodes limited computational capacity in each frame

How to Find A Schedule with Flow Graph

- Maximum attainable flow is $\Sigma_{i=1..N}$ Ci
  - Total amount of computation in the hyperperiod
  - If a max flow is found with this amount then we have a schedule

- If a task is scheduled across multiple frames, we must slice it into subtasks
  - Potentially difficult. However, if we don’t allow the algorithm to split tasks, the problem becomes NP-complete.
  - Analogy: Optimal bin packing becomes easy if we can split objects

Illustration of Job Splitting



Example

Given the task set: T1(4, 1); T2(5, 1); T3(10, 2).

(1) Determine the size of the hyperperiod.

(2) Find suitable frame size(s).

(3) Choose a reasonable frame size, give a cyclic executive schedule of these three tasks in a hyperperiod using network flow model.

Solution

1) Determine the size of the hyperperiod.

H = lcm(4, 5, 10) = 20

(2) Find suitable frame size(s).

C1: f>=2

C2: f can be 2, 4, 5, 10, 20

C3: 2f – gcd (Pi, f) <= Di

f = 2 (OK)

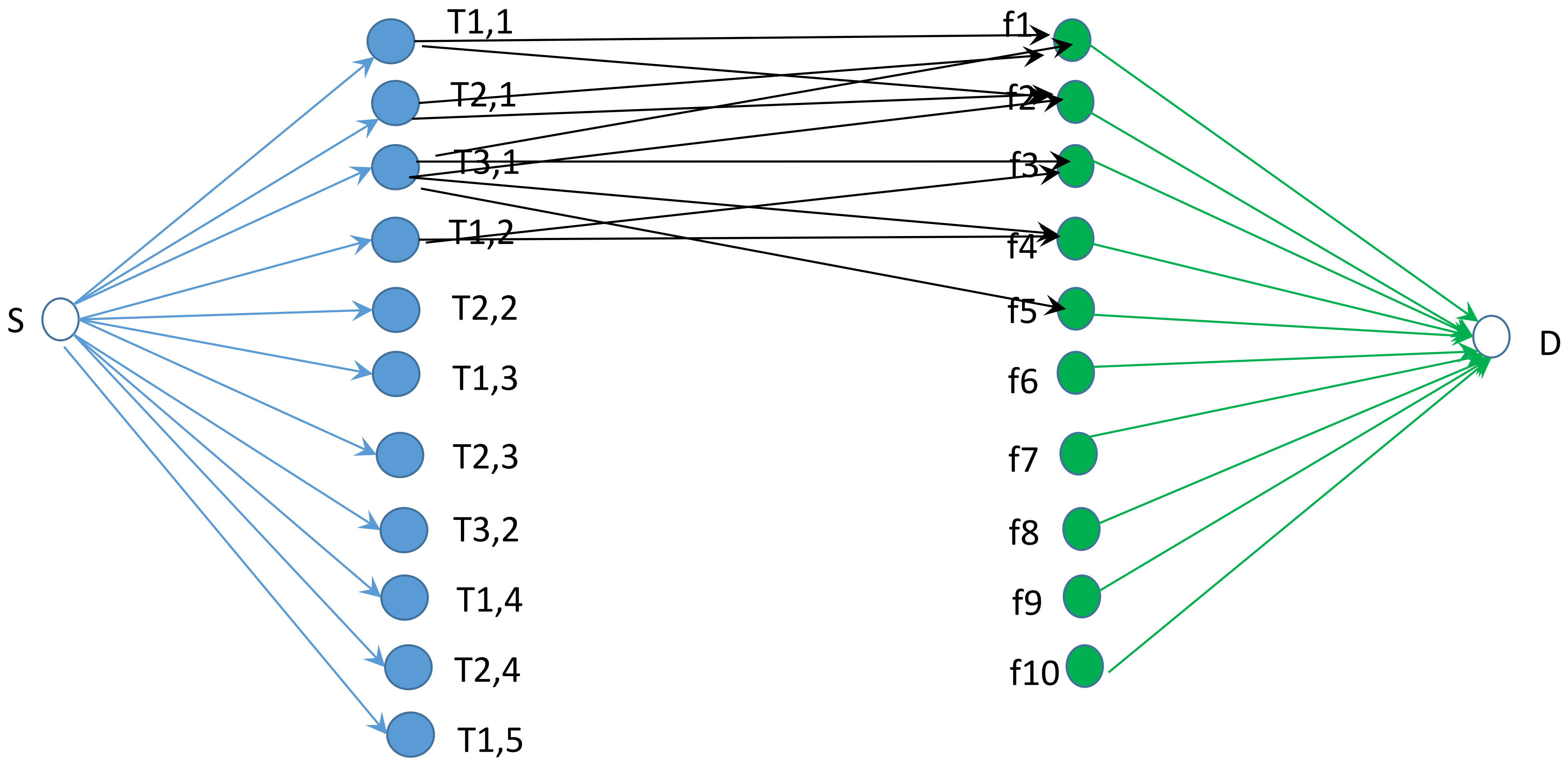
f = 4 T1: 8-4 = 4, T2: 8-1=7>5, so f=4 not suitable

f = 5 T1: 10-1>4, so f=5 not suitable

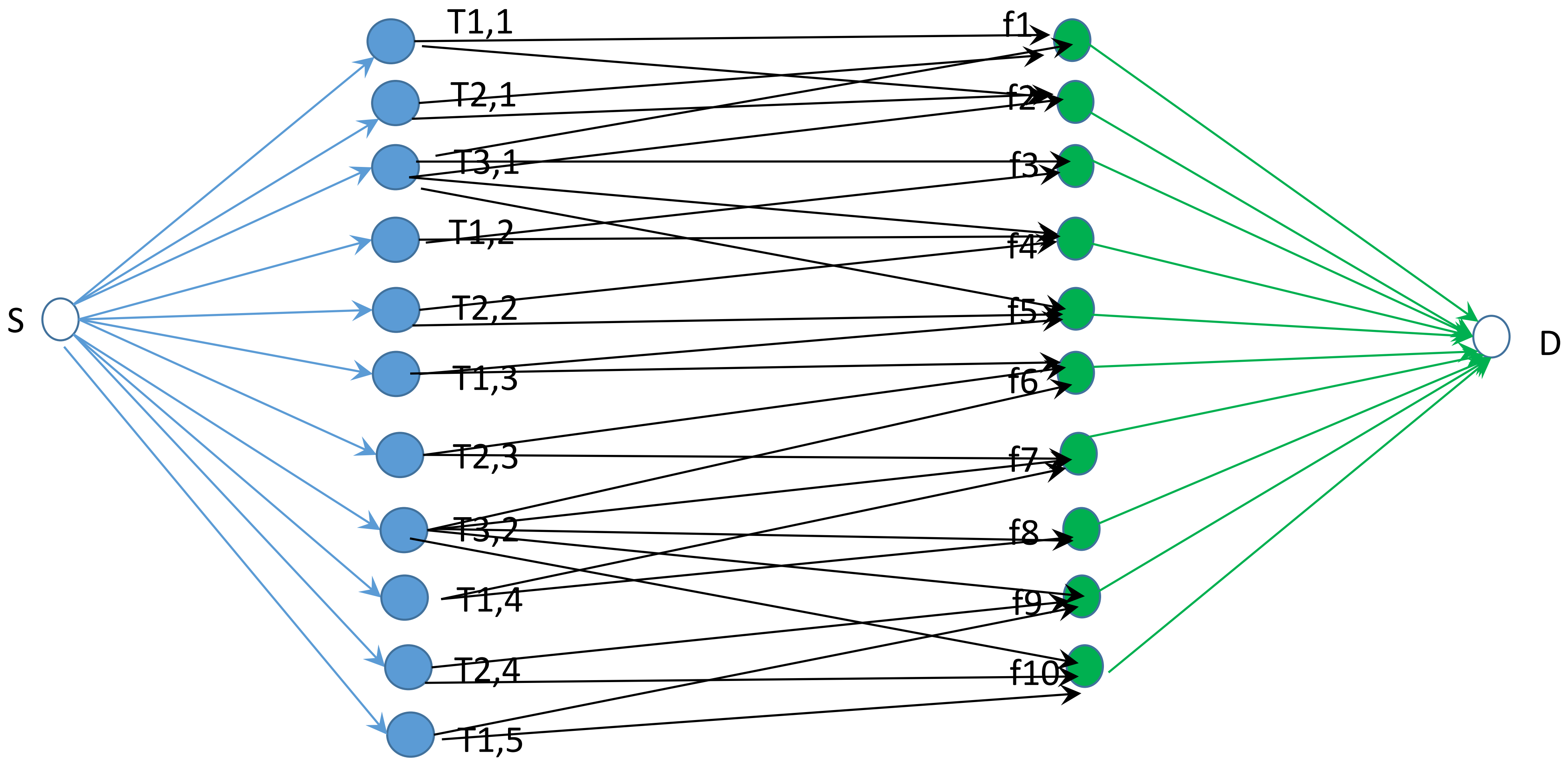
… …

f = 2 is the suitable frame size!

Use network flow to model the CE scheduling



Use network flow to model the CE scheduling



CE Advantages

- Cyclic executives are very simple – you just need a table
- Table makes the system very predictable
  - Can validate and test with very high confidence
- No race conditions, no deadlock
- Task dispatch is very efficient: just a function call

CE Disadvantages

- Cyclic executives are brittle – any change requires a new table to be computed
- The number of frames (F) could be huge
  - Implies mode changes may have long latency
- Release times of tasks must be fixed
- Slicing tasks into smaller units is difficult and error prone

Summary

- Cyclic executive is one of the major software architectures for embedded systems
- Historically, cyclic executives dominate safety-critical systems
- Simplicity and predictability win
- However, there are significant drawbacks (overhead, strong assumptions)
- Finding a schedule might require significant offline computation

End