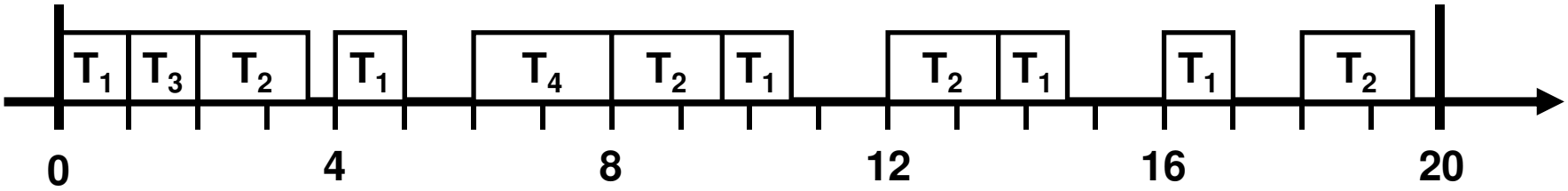
Example

- Consider a system with four tasks
  - $T_1 = (4,1)$
  - $T_2 = (5, 1.8)$
  - $T_3 = (20, 1)$
  - $T_4 = (20, 2)$
- Possible schedule:



- Table starts out with:
  - $(0, T_1), (1, T_3), (2, T_2), (3.8, I), (4, T_1), \ldots$

Refinement: Frames

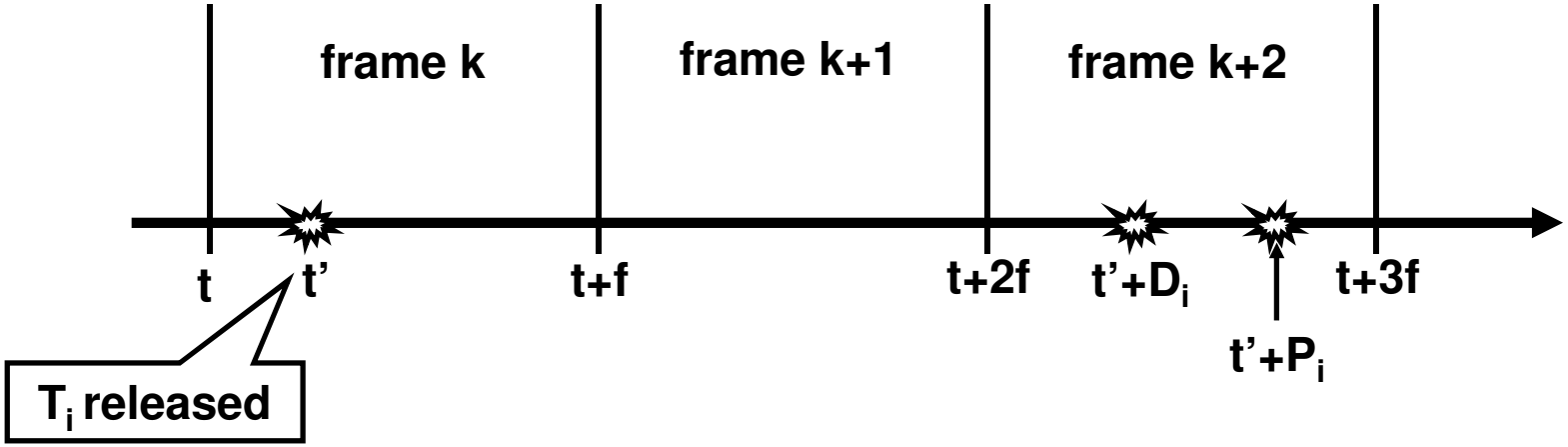
- We divide hyperperiods into *frames*
  - Timing is enforced only at frame boundaries
  - Each task is executed as a function call and must fit within a single frame
  - Multiple tasks may be executed in a frame
  - Frame size is f
  - Number of frames per hyperperiod is F = H/f

Frame Size Constraints

1. **Tasks must fit into frames**
   - **So, $f \geq C_i$ for all tasks**
   - **Justification: Non-preemptive tasks should finish executing within a single frame**
2. **f must evenly divide H**
   - **Equivalently, f must evenly divide P_i for some task i**
   - **Justification: Keep table size small**

More Frame Size Constraints

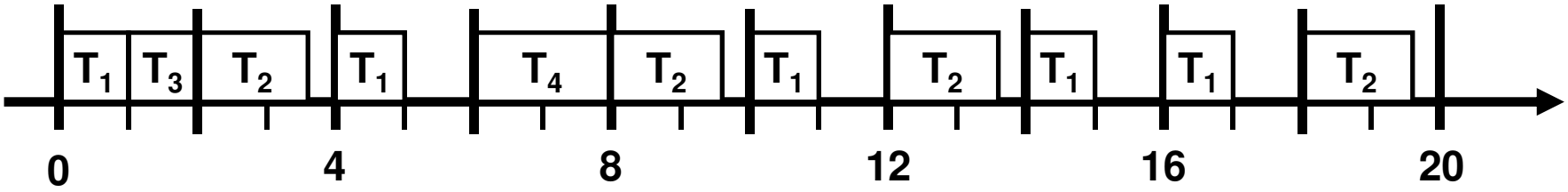
3. **There should be a complete frame between the release and deadline of every task**
   - **Justification: Want to detect missed deadlines by the time the deadline arrives**



   - **Therefore: $2f - \gcd(P_i, f) \leq D_i$ for each task i**

Example Revisited

- Consider a system with four tasks
  - $T_1 = (4,1)$, $T_2 = (5, 1.8)$, $T_3 = (20, 1)$, $T_4 = (20, 2)$
  - H = lcm (4,5,20) = 20
- By Constraint 1: f ≥ 2
- By Constraint 2: f might be 1, 2, 4, 5, 10, or 20
- By Constraint 3: only 2 works



Task Slices

- What if frame size constraints cannot be met?
  - Example: T = { (4, 1), (5, 2, 7), (20, 5) }
    - By Constraint 1: $f \geq 5$
    - By Constraint 3: $f \leq 4$
- Solution: “slice” a task into smaller sub-tasks
  - So (20, 5) becomes (20, 1), (20, 3), and (20, 1)
  - Now f = 4 works
- What is involved in slicing?

Design Decision Summary

- **Three decisions:**
  - **Choose frame size**
  - **Partition tasks into slices**
  - **Place slices into frames**
- **In general these decisions are not independent**

Cyclic Executive Pseudocode

```
// L is the stored schedule
current time t = 0;
current frame k = 0;
do forever
    accept clock interrupt;
    currentBlock = L(k);
    t++;
    k = t mod F;
    if last task not completed, take appropriate action;
    execute slices in currentBlock;
    sleep until next clock interrupt;
```