5.3 Scheduling

- Goals
- *Scheduling Algorithms*
- Implementation Issues
- Case Studies

Scheduling Algorithms

Basic
- FIFO
- SJF
- SRTN
- RR

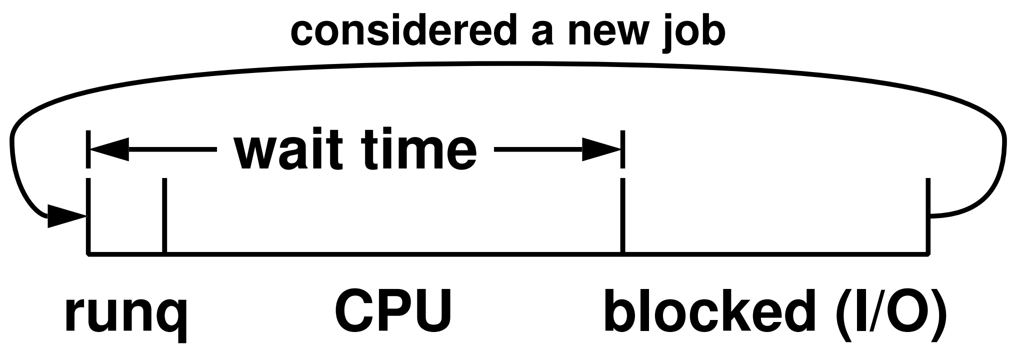
Priority
- Multi-level
- Multi-level w/ Feedback*

Prortional Share
- Lottery
- Stride*

Real Time
- EDF
- Rate Monotonic

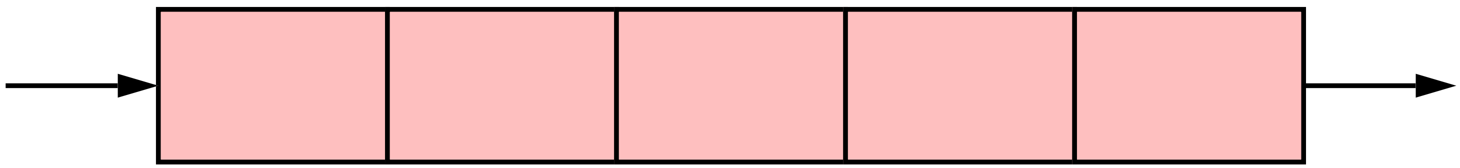
- We will focus on how the *run queue* is managed
- All these schedulers are *work-conserving*
  - if there is work to be perform, must keep CPU busy
  - opposite of "vacationing scheduler"

Scheduling Non-preemptive, Non-interactive Jobs



- Scheduling "jobs"
  - *wait time* is over when job *voluntarily* gives up the CPU
- Run one at a time
  - *no preemption*
- Running time is *known*
  - jobs J_i with *processing/execution time* T_i for $i = 0, 1, 2, \ldots$
  - not realistic
    - *if* you know the running time of every job, what would you do?

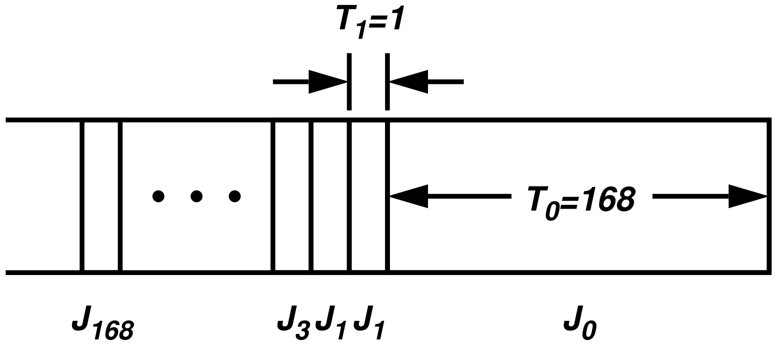
FIFO

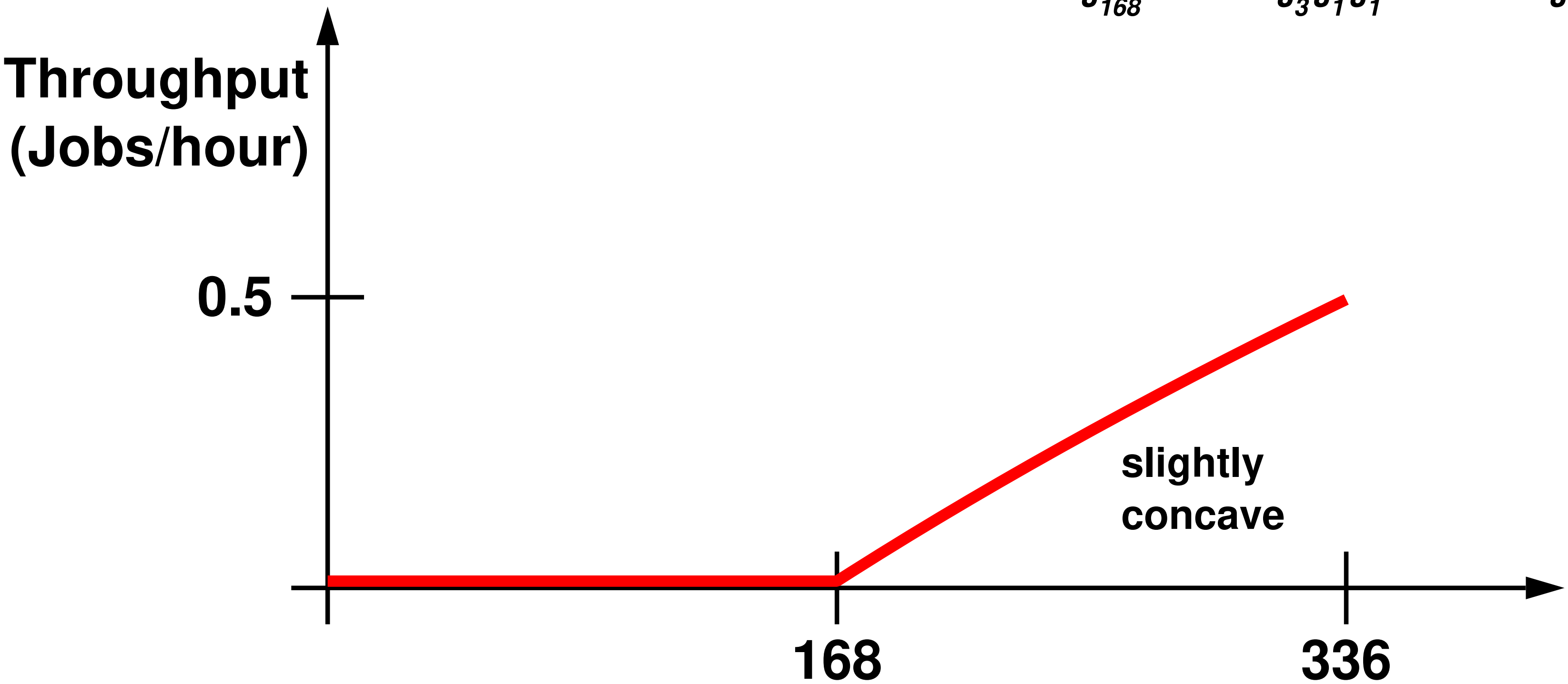


- Ex: weenix

Performance Metric: Throughput

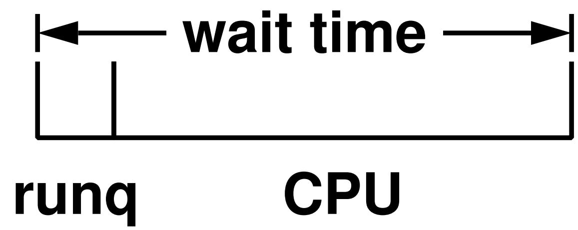
- "Goodness" criterion is jobs/hour
  - assuming that all jobs are sitting in the run queue at time 0
- Ex:
  - one *168*-hour job
  - followed by *168* one-hour jobs





Another Performance Metric: Average Wait Time

- Jobs J_i with *processing/execution time* T_i for $0 \le i < n$
- *Average wait time (AWT)* for FIFO
  - please note that this is *not* the same as any warmup2 "wait time"
  - J_i started at time t_i
  - $t_i = \sum_{j=0}^{i-1} T_j$ (for FIFO)
  - $WT_i = t_i + T_i$ (for non-preemptive schedulers)
  - $AWT = \sum_{i=0}^{n-1} WT_i \,/\, n = \sum_{i=0}^{n-1} (t_i + T_i) \,/\, n$



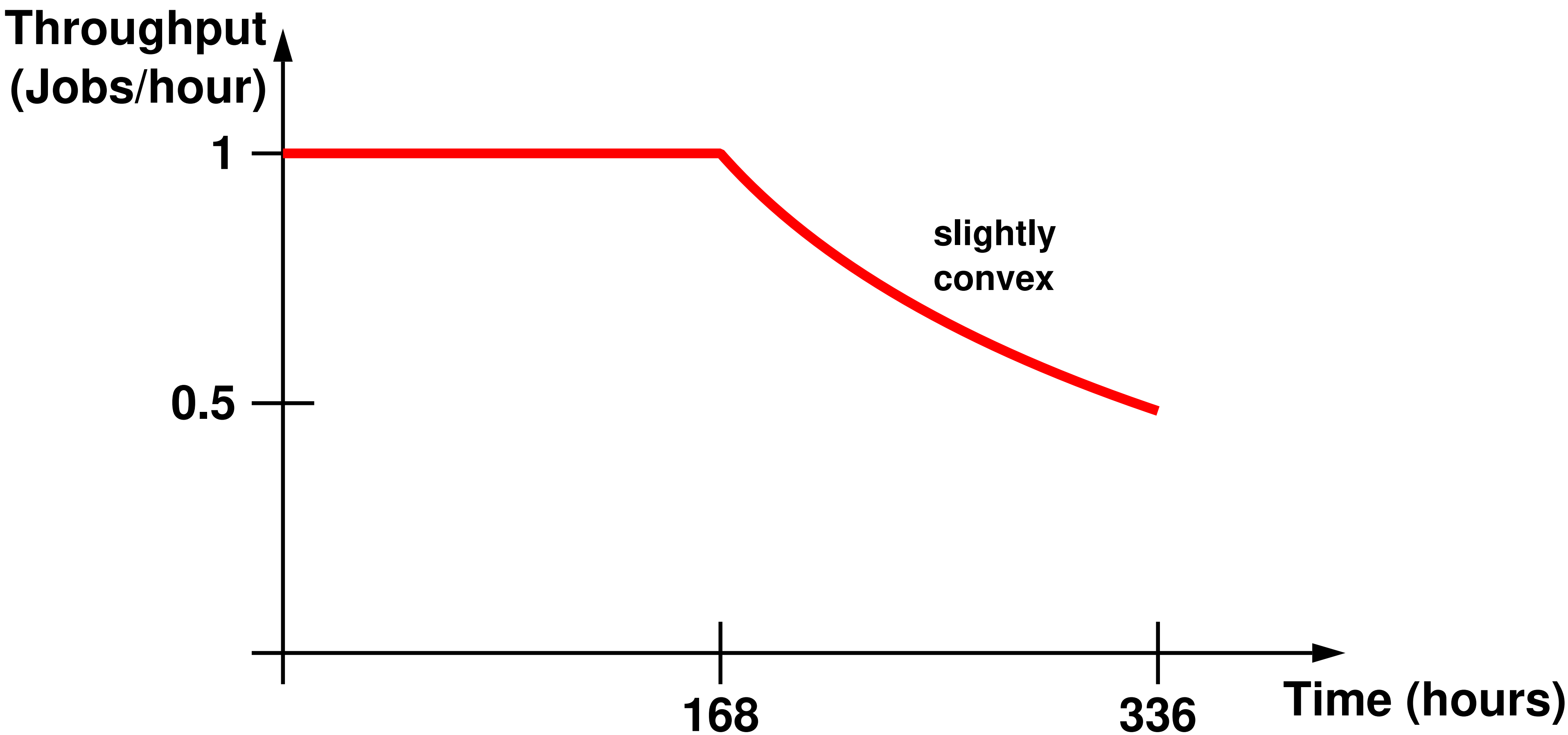
- For our example (which is the *worst-case* for FIFO)
  - AWT = *252* hours (with a standard deviation of *48.79* hours)
    - large average and large variation (for this example)
- In general, AWT for FIFO is more difficult to compute
  - need to look at all possible ordering of jobs and the probability of getting each particular order

Shortest Job First

- How to *minimize* AWT?
- $AWT = \sum_{i=0}^{n-1} (t_i + T_i) / n$
  - $t_i = \sum_{j=0}^{i-1} T_j$

    if $i > 0$, $t_i = T_{i-1} + t_{i-1}$
- $AWT = (nT_0 + (n-1)T_1 + (n-2)T_2 + \ldots + 2T_{n-2} + T_{n-1}) / n$
- Minimized when $T_i \leq T_{i+1}$ for all i
  - which is *Shortest Job First (SJF)*

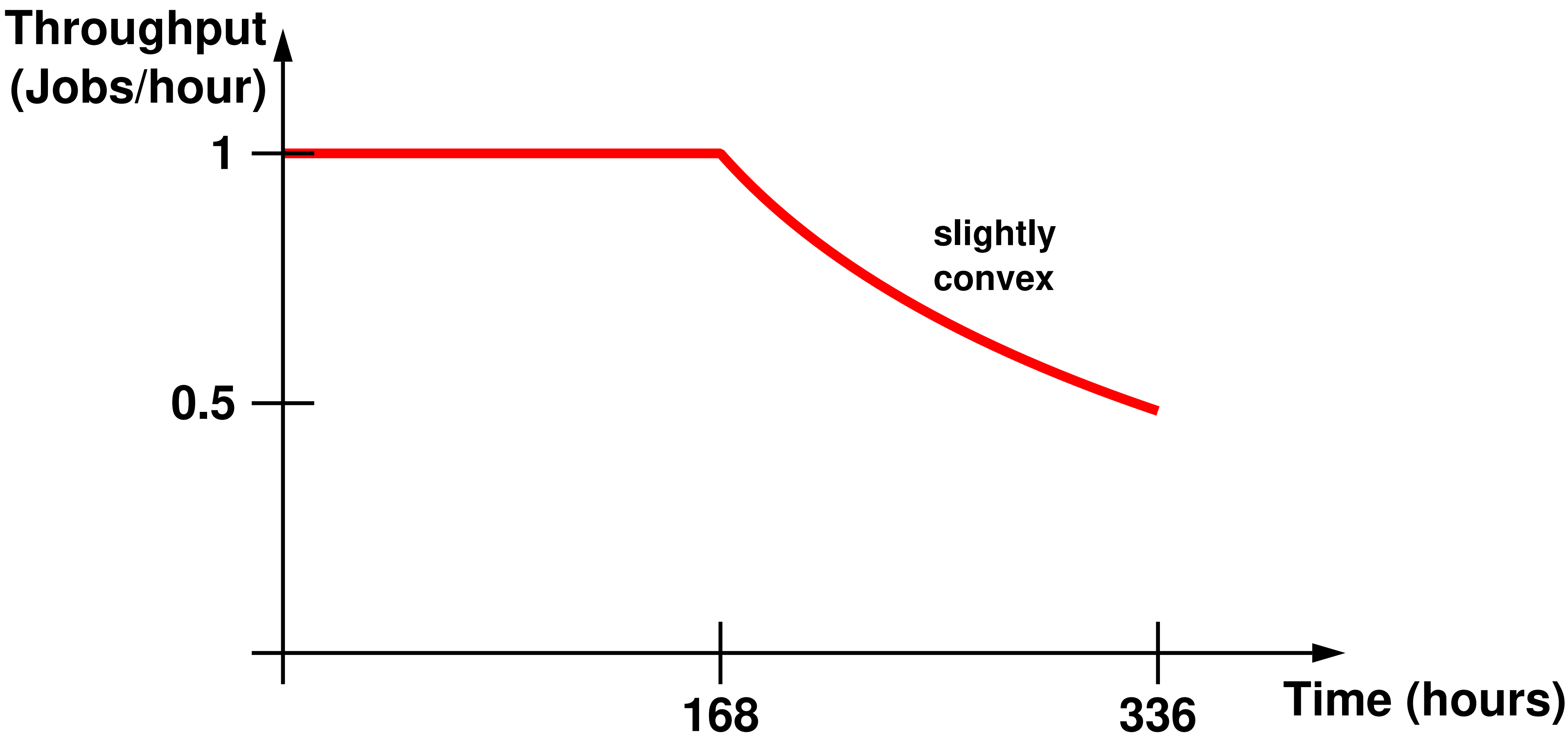
SJF and Our Example



- AWT = *85.99* hours (with a standard deviation of *52.06* hours)

- Instantaneous throughputs can be different for different schedulers
  - but throughput at time = 336 is *identical for all work-conserving schedulers*

SJF and Our Example

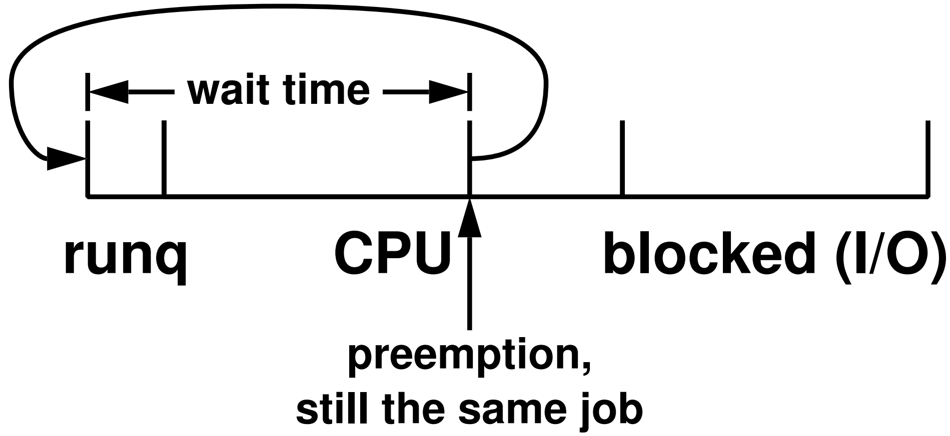


- AWT = *85.99* hours (with a standard deviation of *52.06* hours)

What if short jobs keep arriving?

- *starvation*
  - *unacceptable* for a scheduling policy

Scheduling Preemptive, Non-interactive Jobs



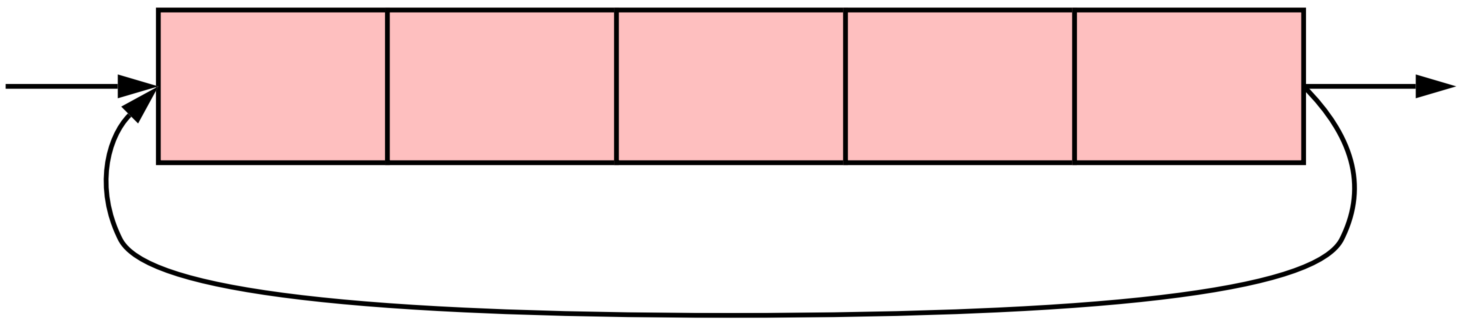
- ***Preemption:***
  - current job may be preempted by others
    - if it goes to the runq due to preemption, it's still the same job
    - it doesn't have to go to the tail of the runq
    - wait time is the time when this job voluntarily gives up the CPU minus the time it first entered the runq
  - to minimize AWT, use ***shortest remaining time next (SRTN)***
    - basically SJF
      - same argument that SJF minimizes AWT when there is no preemption
    - we reserve the term "SJF" to refer to the non-preemptive case

Fairness

- FIFO
  - each job eventually gets processed
  - that seems fair
- SJF and SRTN
  - a long job might have to wait indefinitely
- What's a good measure of fairness?

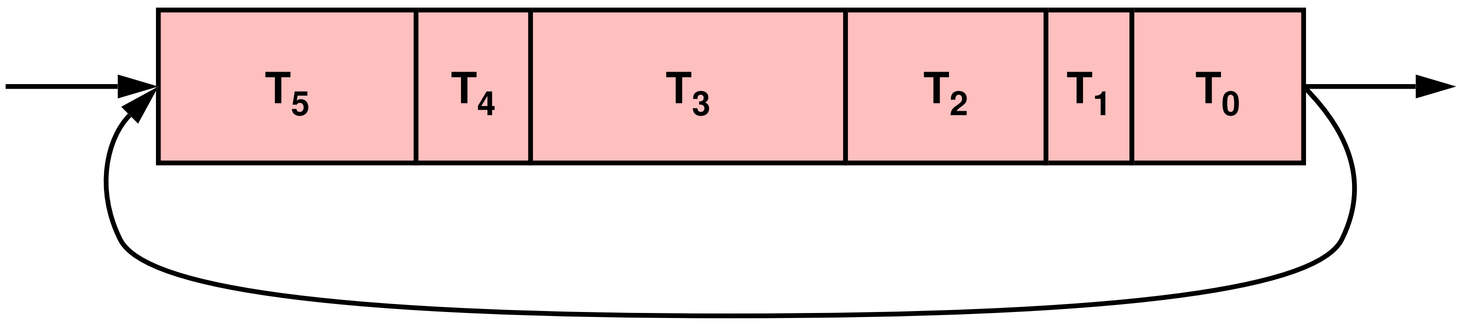
Round Robin

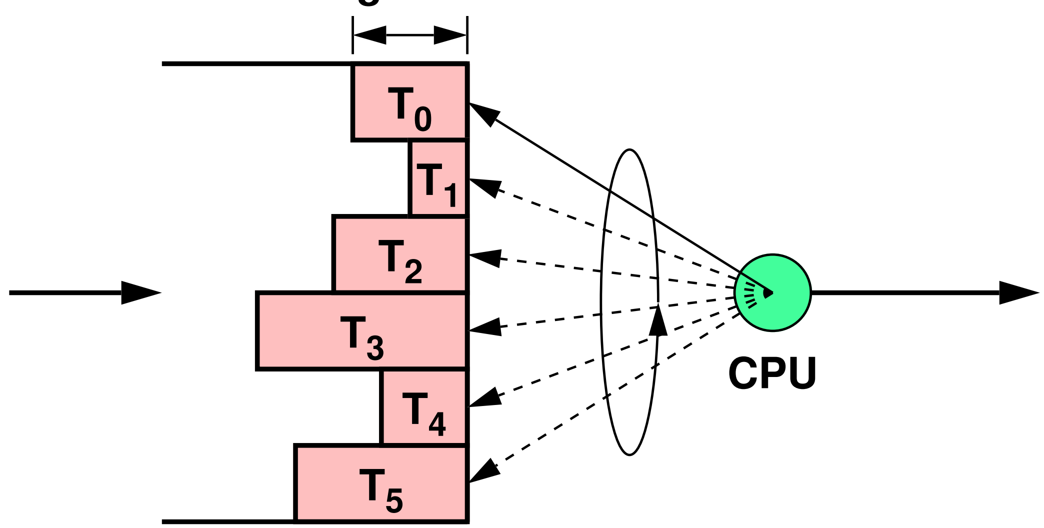
- Time-slicing
  - *q* = quantum or time slice



Round Robin

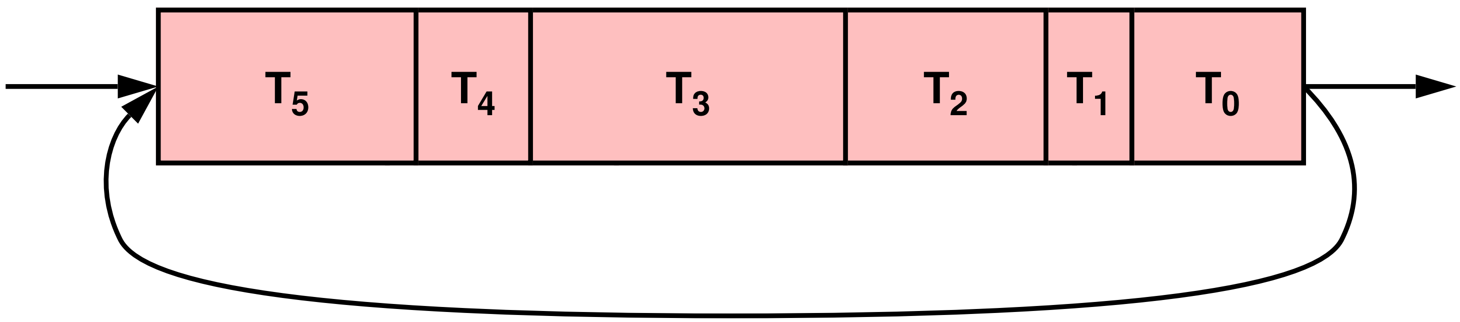
- Time-slicing
  - *q* = quantum or time slice





Round Robin

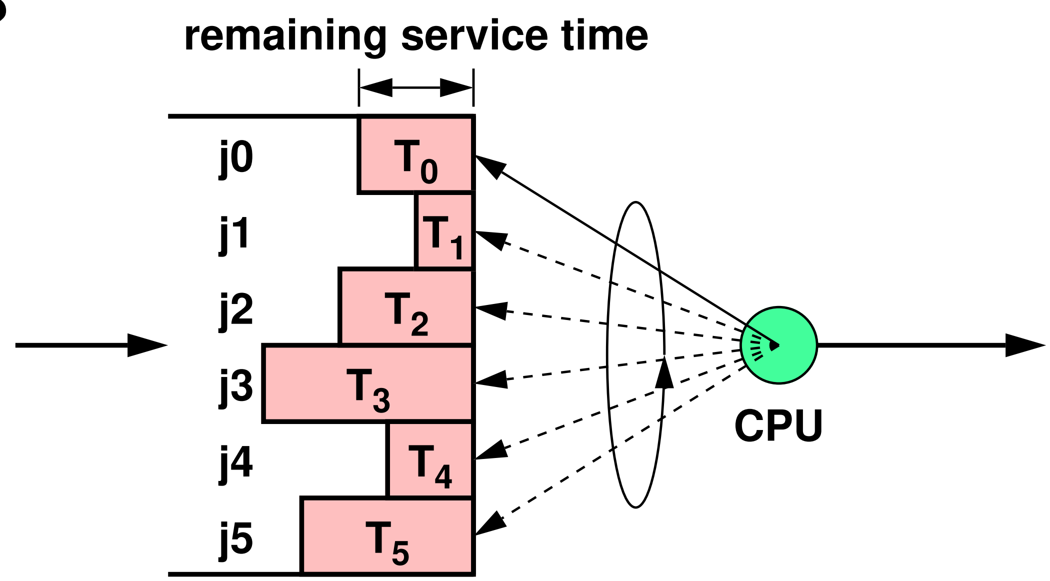
- **Time-slicing**
  - *q* = quantum or time slice



- **Different values of *q***
  - $q \rightarrow 0$: processor-sharing (idealized case)
    - not realistic
    - *translation lookaside buffer flushing* and *caching problem*
      - not enough time to achieve good hit-rate in TLB
  - *q* too large: some jobs appear to be not making progress

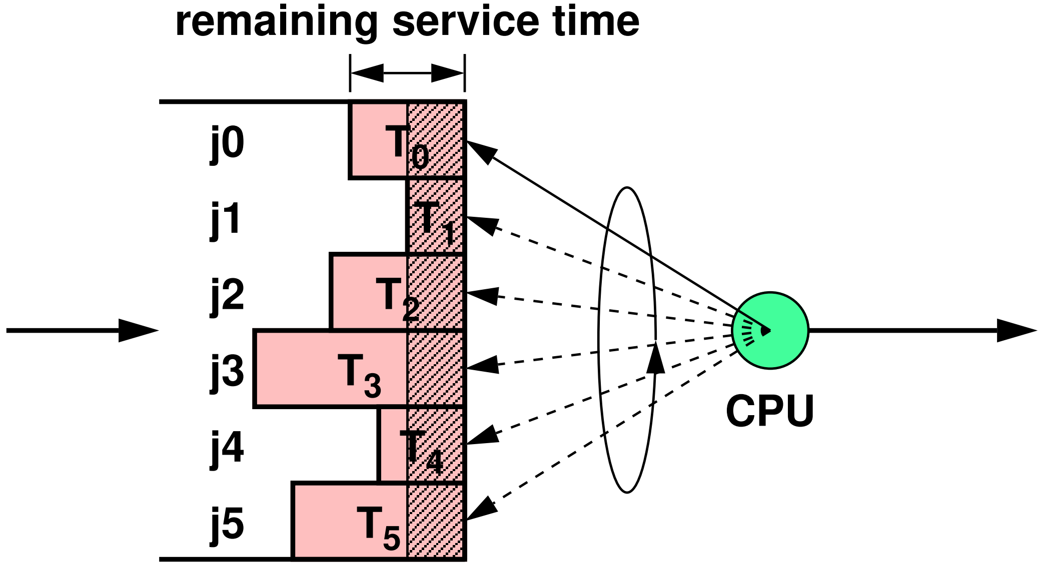
Round Robin

- **How to calculate waiting time?**
  - ***WT_1 =***



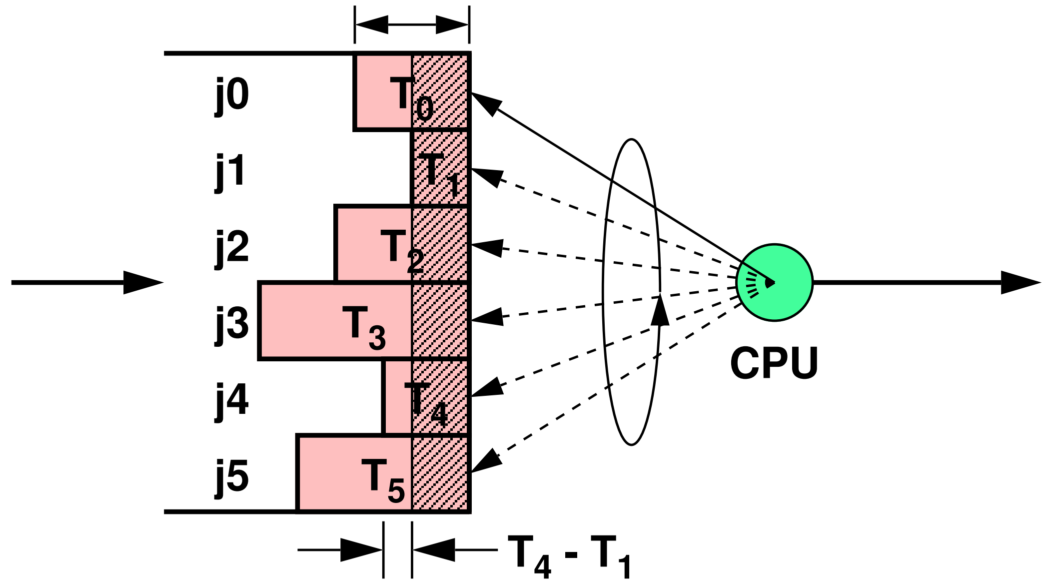
Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
    - why not $6 \times T_1 - 4 \times q$
    - $q \rightarrow 0$, we are doing calculus, not algebra



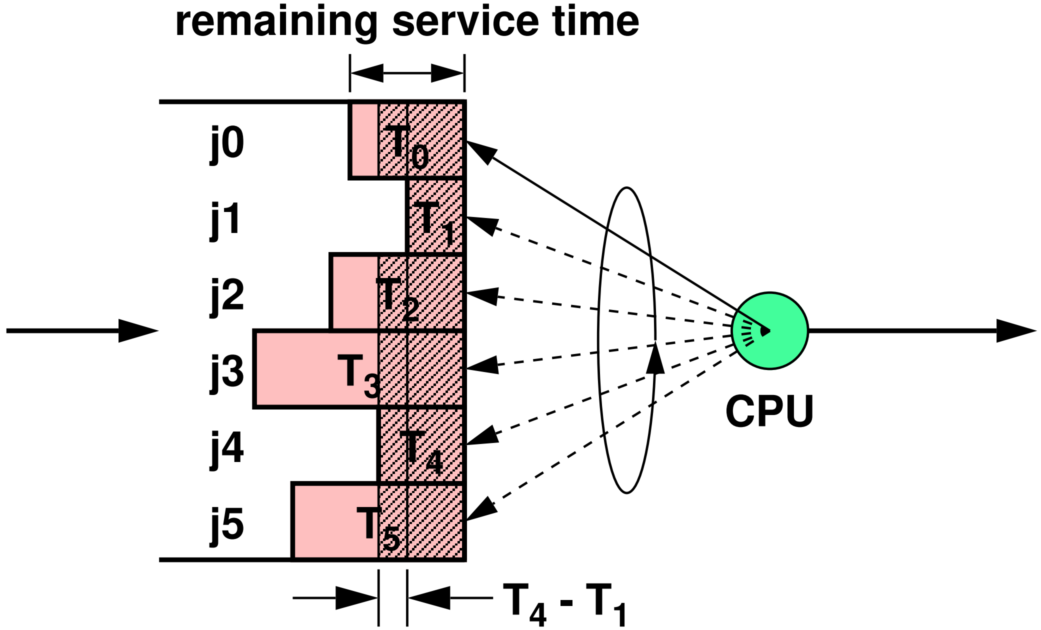
Round Robin

- **How to calculate waiting time?**
  - $WT_1 = 6 \times T_1$
  - $WT_4 = ?$



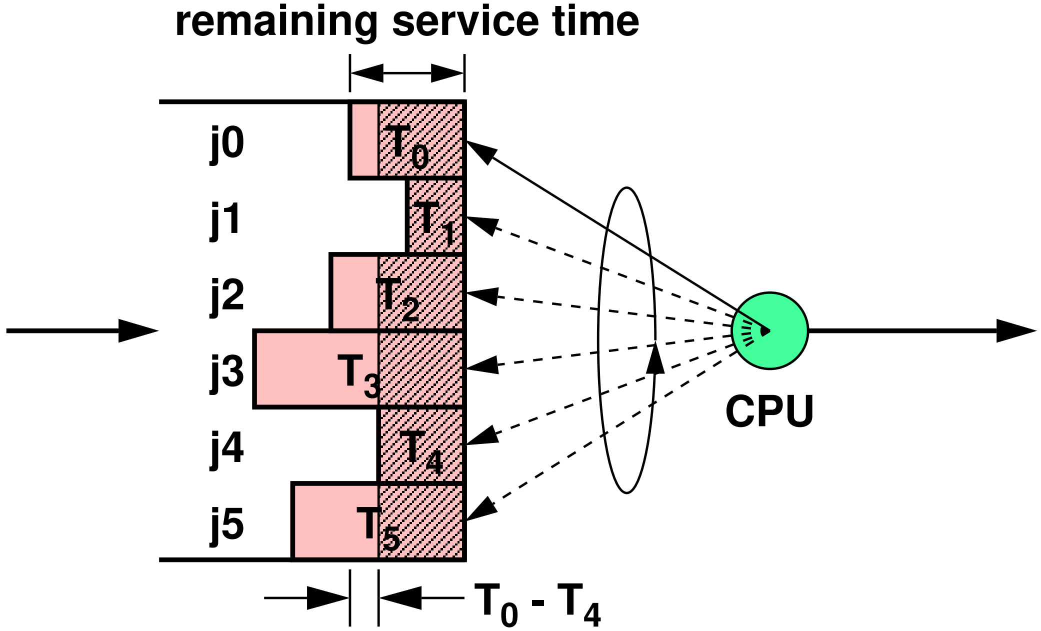
Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$



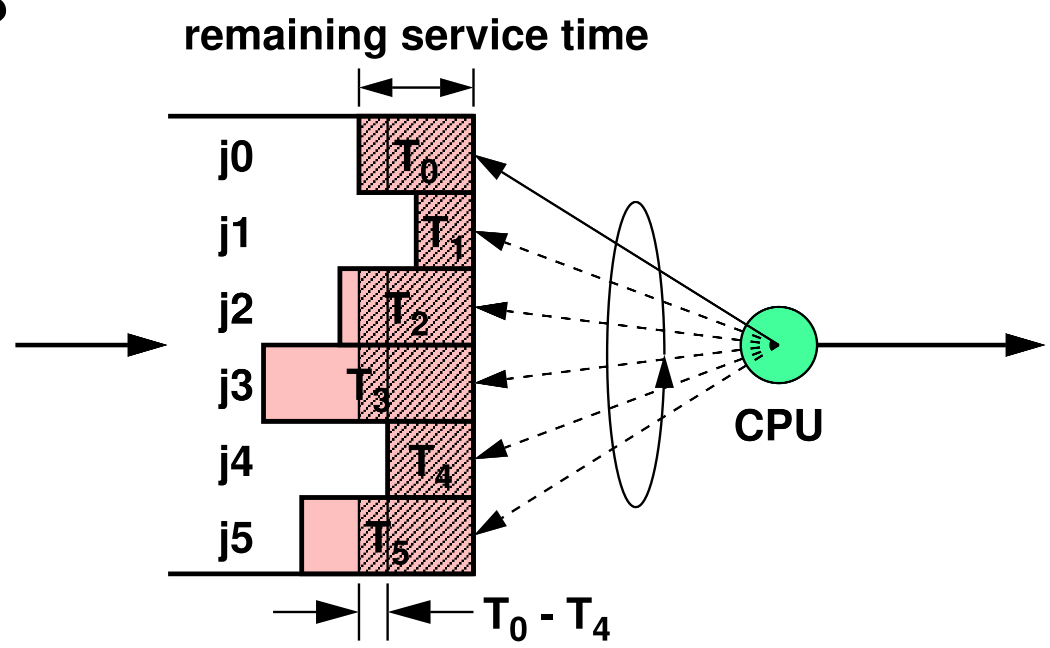
Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = ?$



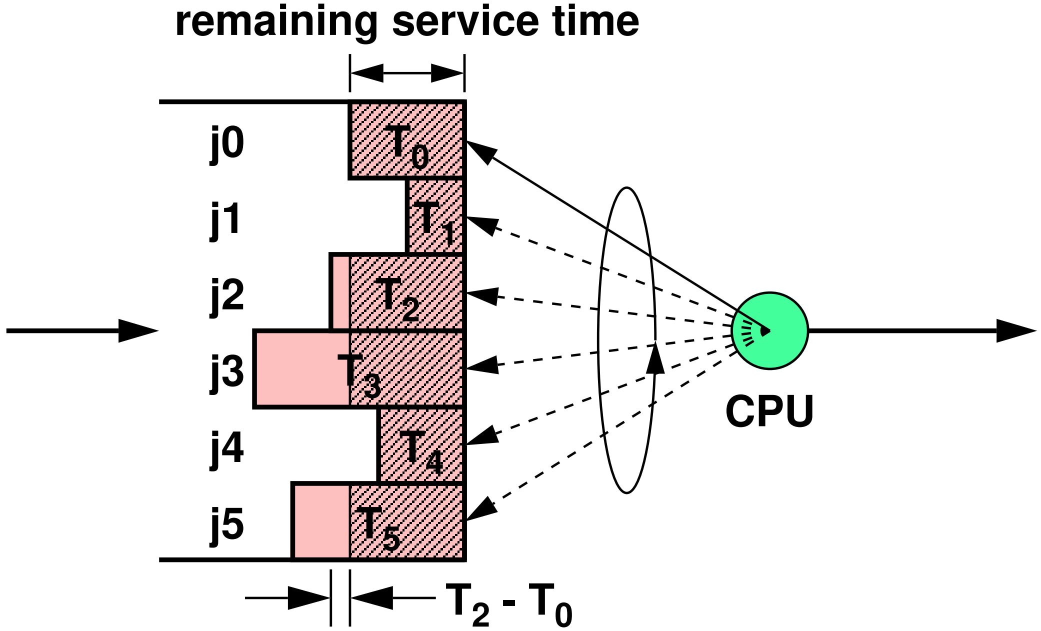
Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$



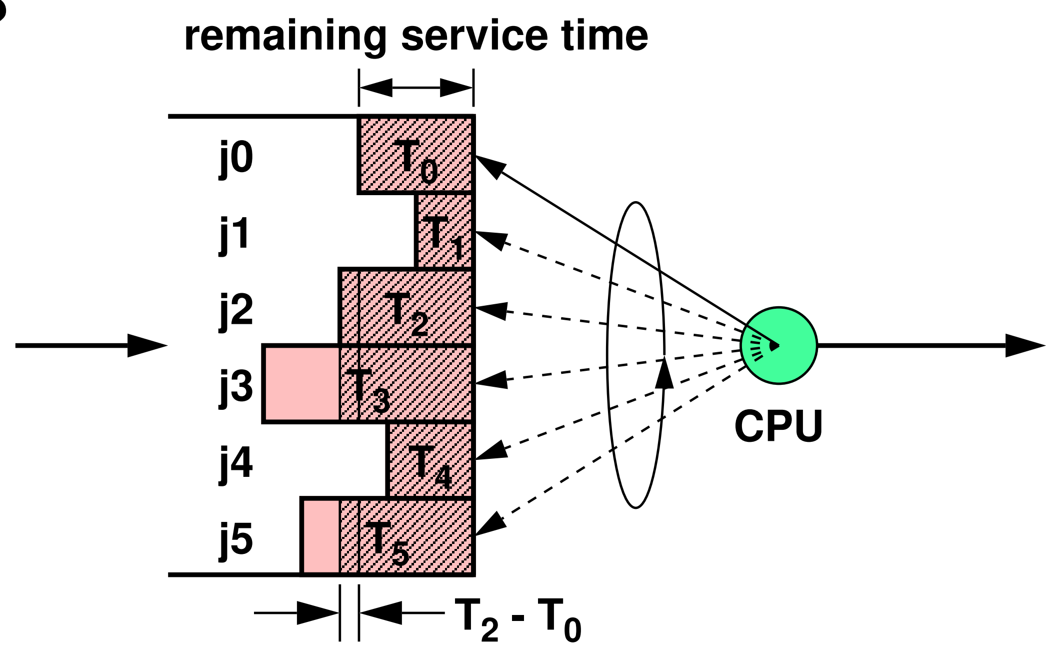
Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$
  - $WT_2 = ?$



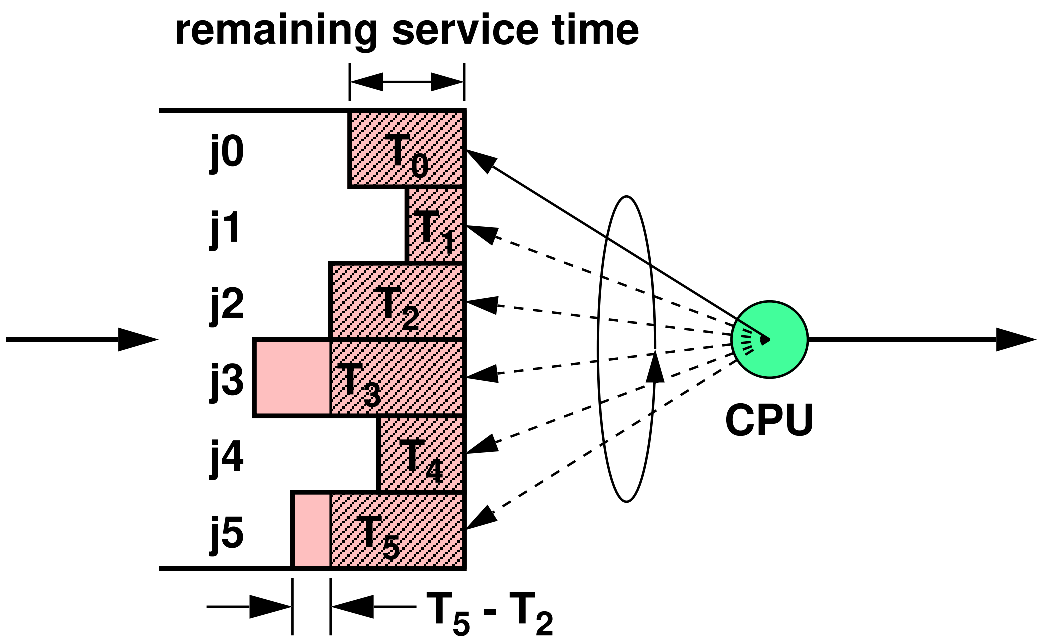
Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$
  - $WT_2 = 3 \times (T_2 - T_0) + WT_0$



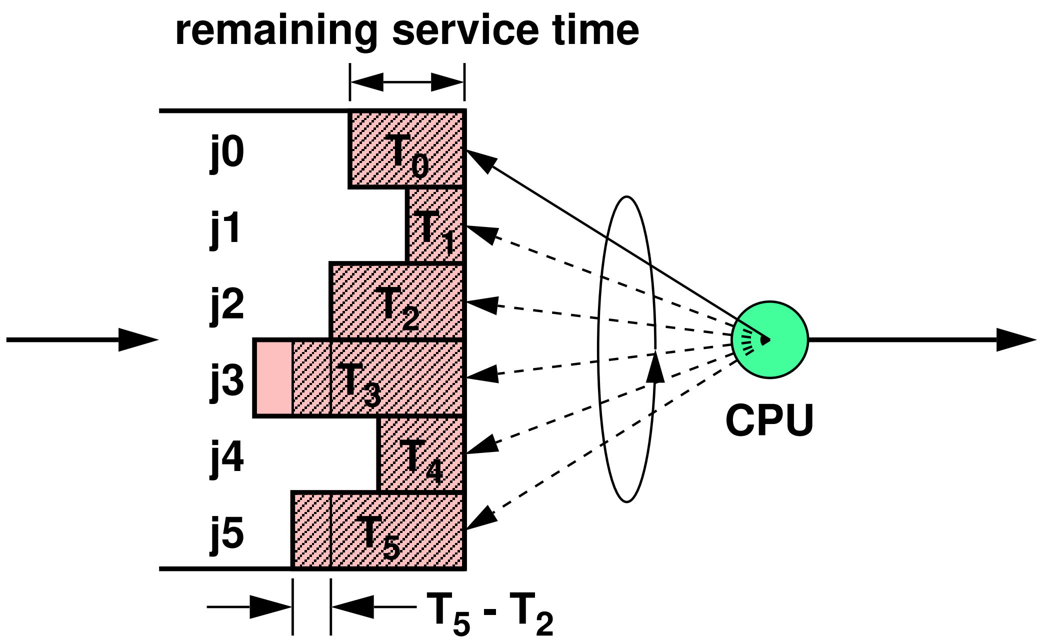
Round Robin

- **How to calculate waiting time?**
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$
  - $WT_2 = 3 \times (T_2 - T_0) + WT_0$
  - $WT_5 = ?$



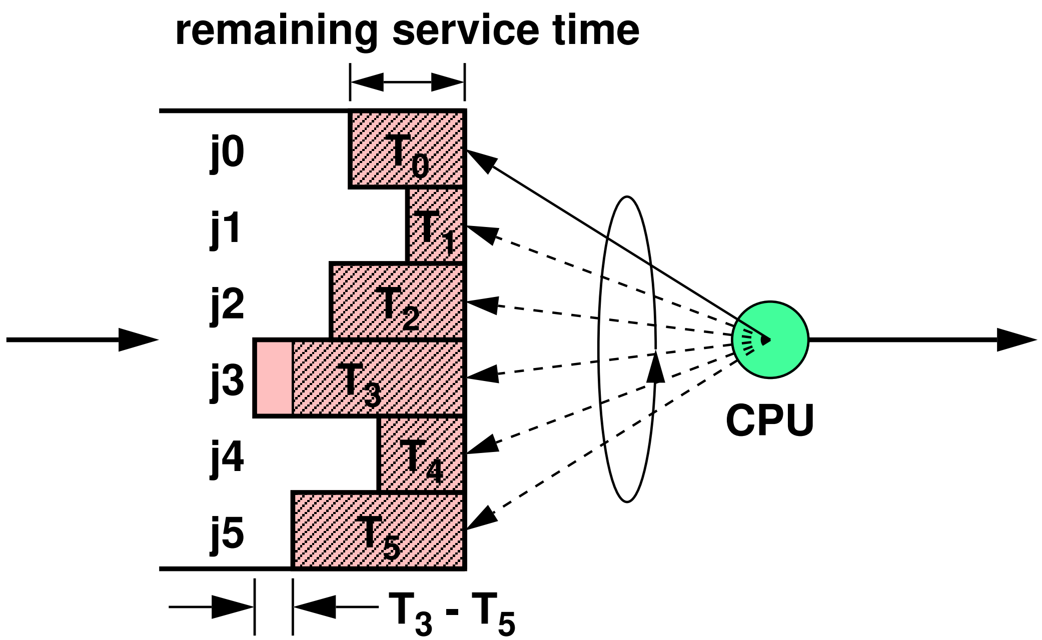
Round Robin

- **How to calculate waiting time?**
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$
  - $WT_2 = 3 \times (T_2 - T_0) + WT_0$
  - $WT_5 = 2 \times (T_5 - T_2) + WT_2$



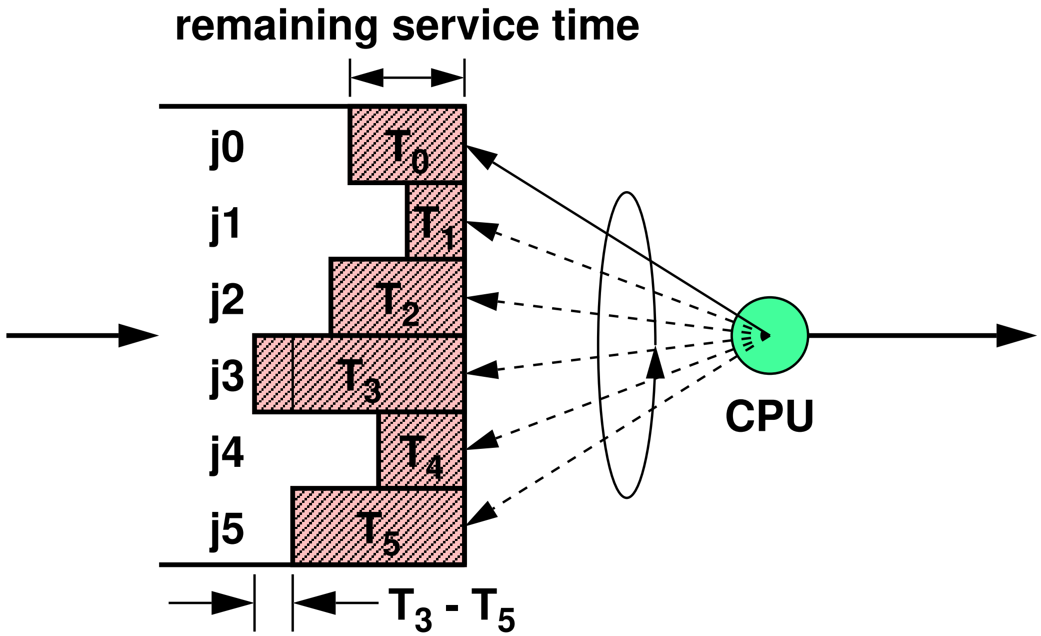
Round Robin

- **How to calculate waiting time?**
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$
  - $WT_2 = 3 \times (T_2 - T_0) + WT_0$
  - $WT_5 = 2 \times (T_5 - T_2) + WT_2$
  - $WT_3 = ?$



Round Robin

- How to calculate waiting time?
  - $WT_1 = 6 \times T_1$
  - $WT_4 = 5 \times (T_4 - T_1) + WT_1$
  - $WT_0 = 4 \times (T_0 - T_4) + WT_4$
  - $WT_2 = 3 \times (T_2 - T_0) + WT_0$
  - $WT_5 = 2 \times (T_5 - T_2) + WT_2$
  - $WT_3 = 1 \times (T_3 - T_5) + WT_5$



- Does it check out?
  - $WT_3 = T_0 + T_1 + T_2 + T_3 + T_4 + T_5$

Round Robin + FIFO

- AWT?
  - let quantum approach *0* (and pretend that it’s realistic)
  - *169* jobs sharing the processor
  - run at *1/169th* speed for first week
  - short jobs receive one hour of processor time in *169* hours
    - different from SJF since all short jobs finish at about the same time
  - long job completes in *336* hours
  - AWT = *169.99* hours
    - recall that AWT is *252* hours for FIFO in our example and *85.99* hours for SJF
  - *average deviation = 12.81 hours*
    - recall that average deviation = *48.79* hours for FIFO in our example and *52.06* hours for SJF
- RR + FIFO appears to be "fair"
  - no starvation