第15 - 16讲 进程调度算法

§2.4 Scheduling Algorithms

Decision Mode

- ***Nonpreemptive(非剥夺方式)***
  - **Once a process is in the *running* state, it will continue until it terminates or blocks itself for I/O.**
  - **主要用于批处理系统。**
- ***Preemptive(剥夺方式)***
  - **Currently running process may be interrupted and moved to the *Ready* state by the operating system**

Decision Mode

- *Preemptive(剥夺方式)*
  - **Allows for better service since any one process cannot monopolize the processor for very long.**
  - **主要用于**实时**性要求**较**高的**实时**系**统**及性能要求**较**高的批**处**理系**统**和分**时**系**统。

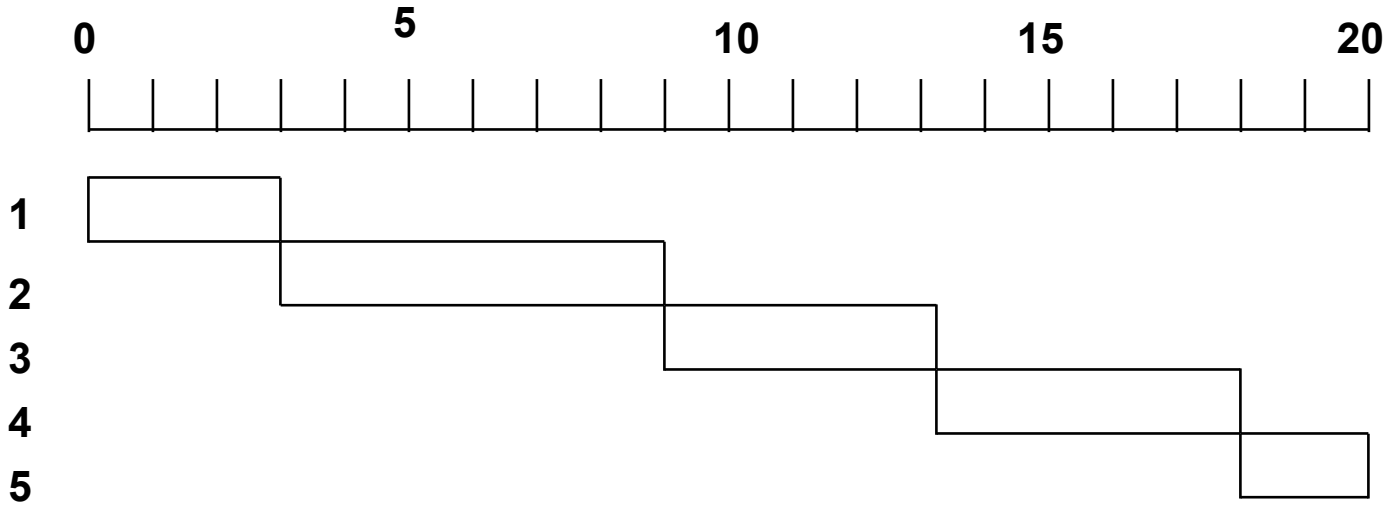
Process Scheduling Example

例如，有 5 个进程，描述如下：

进程	到达时刻	服务时间
P1	0	3
P2	2	6
P3	4	4
P4	6	5

以下部分介绍用 6 种不同的短程调度算法调度这 5 个进程，比较各种*调度算法的平均周转时间*。分析各种调度算法的优缺点。

First-Come-First-Served (FCFS)



- Each process joins the Ready queue.
- When the current process ceases to execute, the oldest process in the Ready queue is selected.

First-Come-First-Served (FCFS)

- 计算进程的平均周转时间
- A short process may have to wait a very long time before it can execute.
- Favors CPU-bound processes.
  - I/O processes have to wait until CPU-bound process completes.

FCFS is unfair to short process

Process	Arrival Time	Service Time (T_s)	Start Time	Finish Time	Turn around Time (T_q)	T_q/T_s
A	0	1	0	1	1	1
B	1	100	1	101	100	1
C	2	1	101	102	100	100
D	3	100	102	202	199	1.99
Mean					100	26

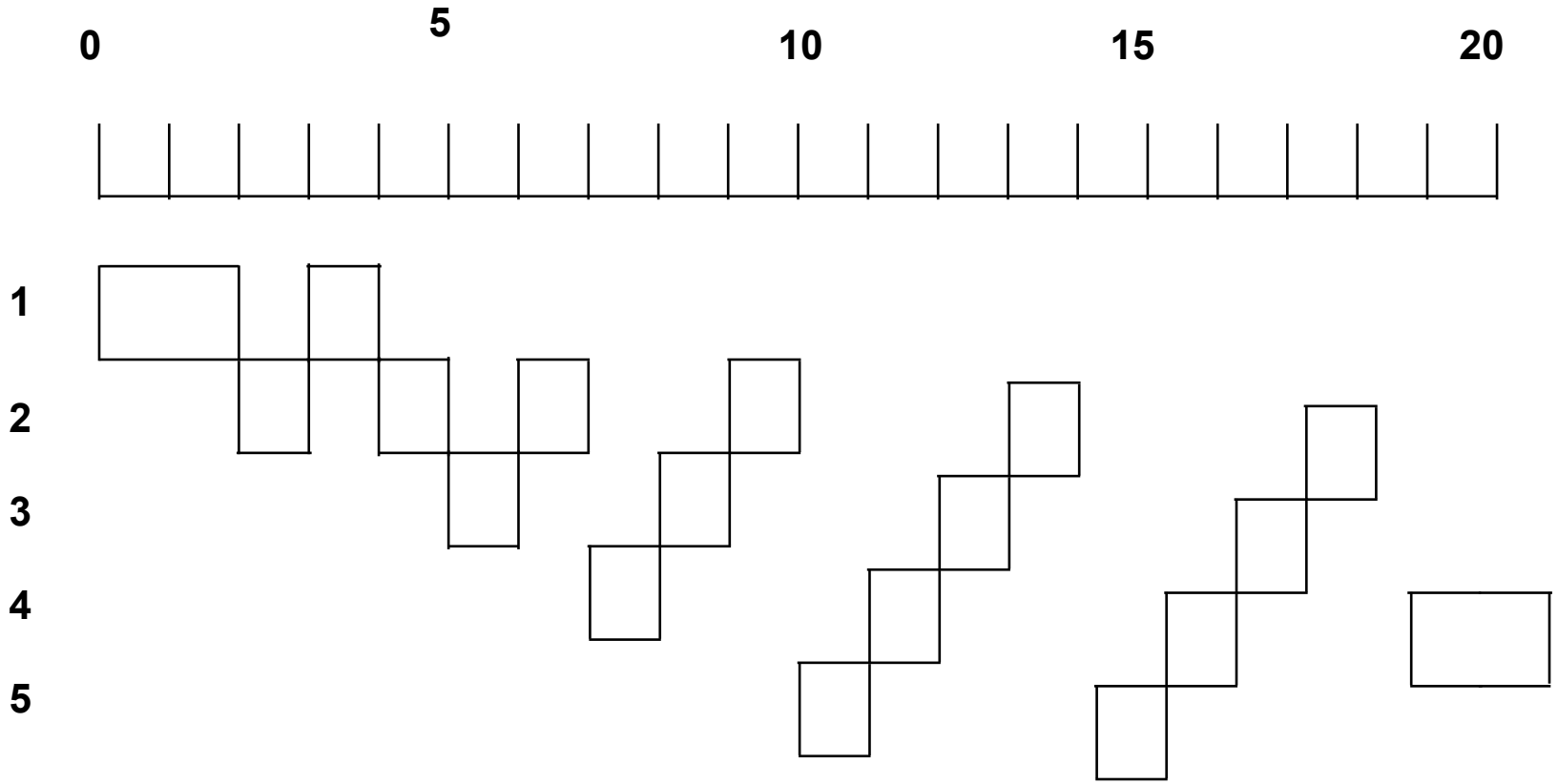
FCFS 算法的实际应用

- 一般， FCFS 与其他调度算法混合使用。
- 系统可以按照不同的优先级维护多个就绪队列，每个队列内部按照 FCFS 算法调度。

Round Robin （RR，轮转法，时间片法）

- Uses preemption based on a clock.
- An amount of time is determined that allows each process to use the processor for that length of time.
  - Known as time slicing(时间片).
- Clock interrupt is generated at periodic intervals.
- When an interrupt occurs, the currently running process is placed in the Ready queue.
  - Next ready job is selected

Round Robin （RR，轮转法，时间片法）



Round Robin （RR，轮转法，时间片法）

RR 算法分析：

1. 时间片的大小对计算机性能的影响

2. 存在的问题：

-未有效利用系统资源。 Processor-bound 进程充分利用时间片，而 I/O- bound 进程在两次 I/O 之间仅需很少的 CPU 时间，却需要等待一个时间片。

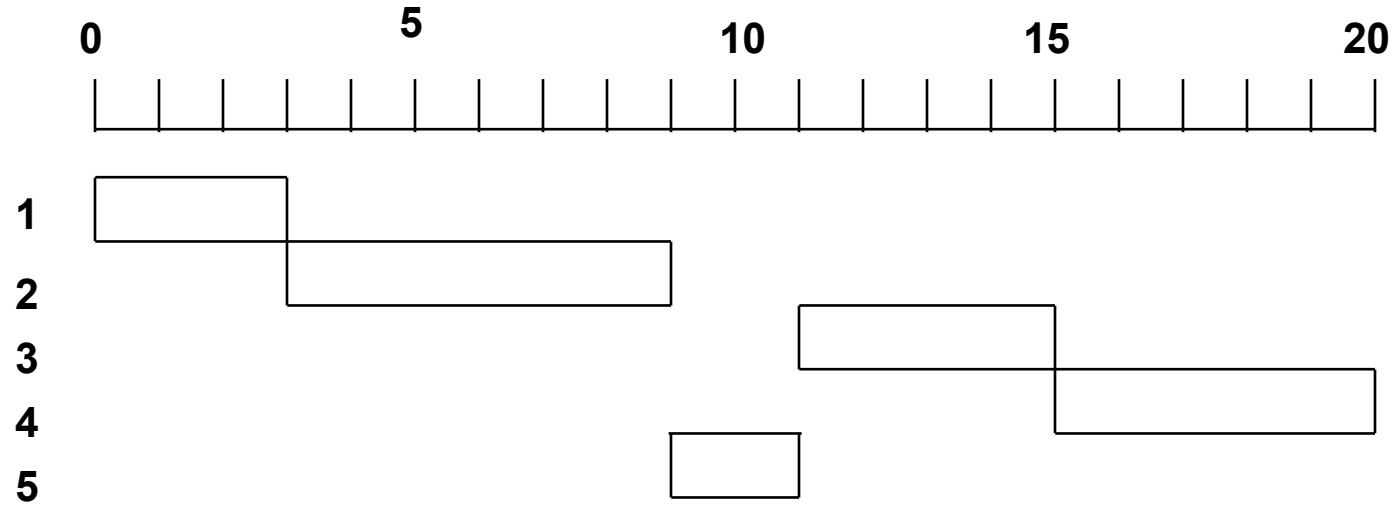
-如何改进？

Virtual round robin （VRR）

- RR 算法的改进：增加一个辅助队列，接收 I/O 阻塞完成的进程，调度优先于就绪队列，但占用的处理机时间小于就绪队列的时间片。
- 算法分析： VRR 算法比 RR 算法公平。
- RR 算法常用于分时系统及事务处理系统。

Shortest Process Next (SPN, 短进程优先)

- Policy: Preemptive or Nonpreemptive?
- Process with shortest expected processing time is selected next.
- Short process jumps ahead of longer processes.



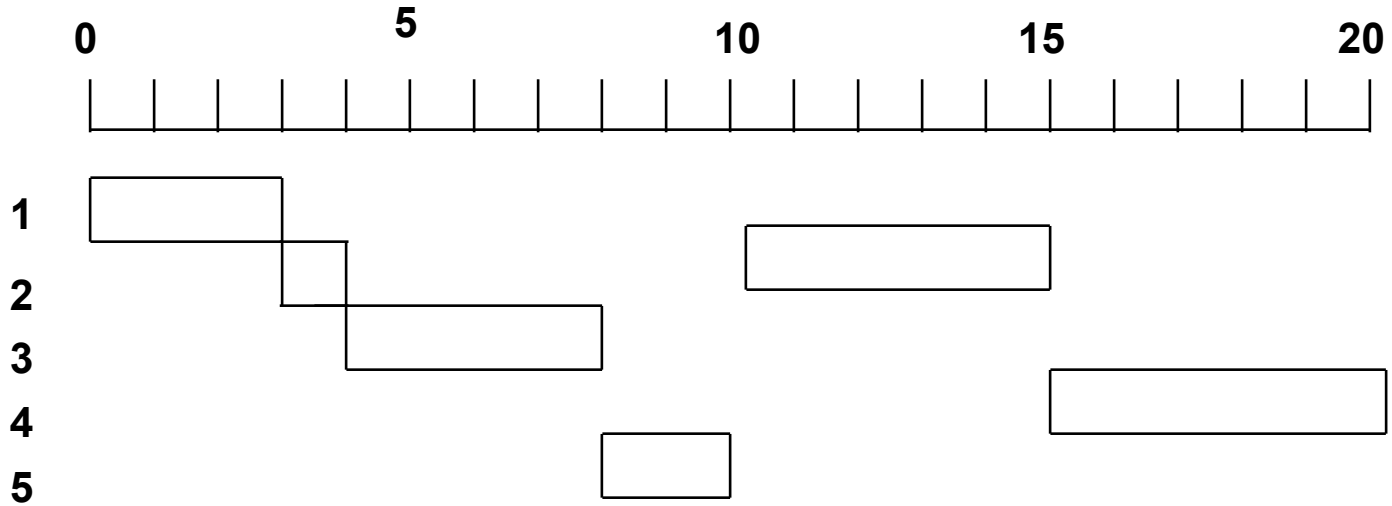
Shortest Process Next (SPN, 短进程优先)

SPN 算法分析：

1. 改善了系统的性能，降低了系统的平均等待时间，提高了系统的吞吐量
2. 存在的问题：
   - 很难准确确定进程的执行时间
   - 对长进程不公平， Possibility of starvation for longer processes
   - 未考虑进程的紧迫程度，不适合于分时系统和事务处理系统

Shortest Remaining Time (SRT，剩余时间最短者优先)

- Preemptive version of shortest process next policy.
- Must estimate processing time.



Shortest Remaining Time (SRT，剩余时间最短者优先)

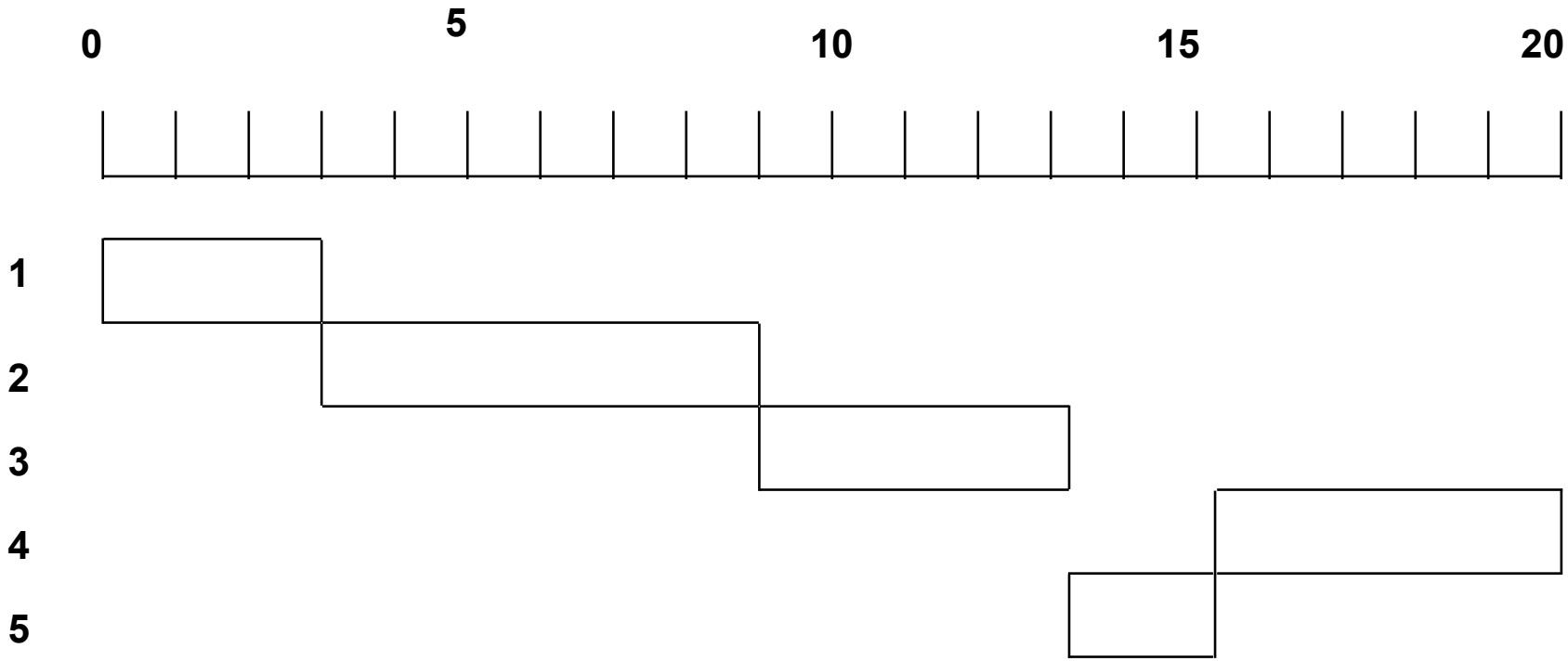
SRT 算法分析：

1. 与 SPN 算法一样，很难准确确定进程的剩余执行时间，且对长进程不公平
2. 但是，它不象 FCFS 算法偏袒长进程，也不象 RR 算法会产生很多中断而增加系统负担。
3. 由于短进程提前完成，故其平均周转时间比 SPN 短。

Highest Response Ratio Next (HRRN, 响应比高者优先)

- 响应比 RR = (time spent waiting + expected service time) / expected service time.
- Choose next process with the lowest response ratio.

Highest Response Ratio Next (HRRN, 响应比高者优先)



Highest Response Ratio Next (HRRN, 响应比高者优先)

HRRN 算法分析：

1. 若进程的等待时间相同，则要求服务时间短的进程优先 （ SPN ）；
2. 若进程要求的服务时间相同，则等待时间长的进程优先（ FCFS ）；
3. 长进程随着等待时间增加，必然会被调度。
4. 但是，很难准确确定进程要求的执行时间，且每次调度之前需要计算响应比，增加系统开销

Feedback (FB，反馈调度法)

1. 设置多个就绪队列，各队的优先权不同（从上而下优先级逐级降低）；
2. 各个队列中进程执行的时间片不同，优先权愈高时间片愈短；
3. 新进程首先放入第一队列尾，按 FCFS 方法调度，若一个时间片未完成，到下一队列排队等待；
4. 仅当第一队列空时，调度程序才开始调度第二队列的进程，依次类推。

Feedback (FB，反馈调度法)

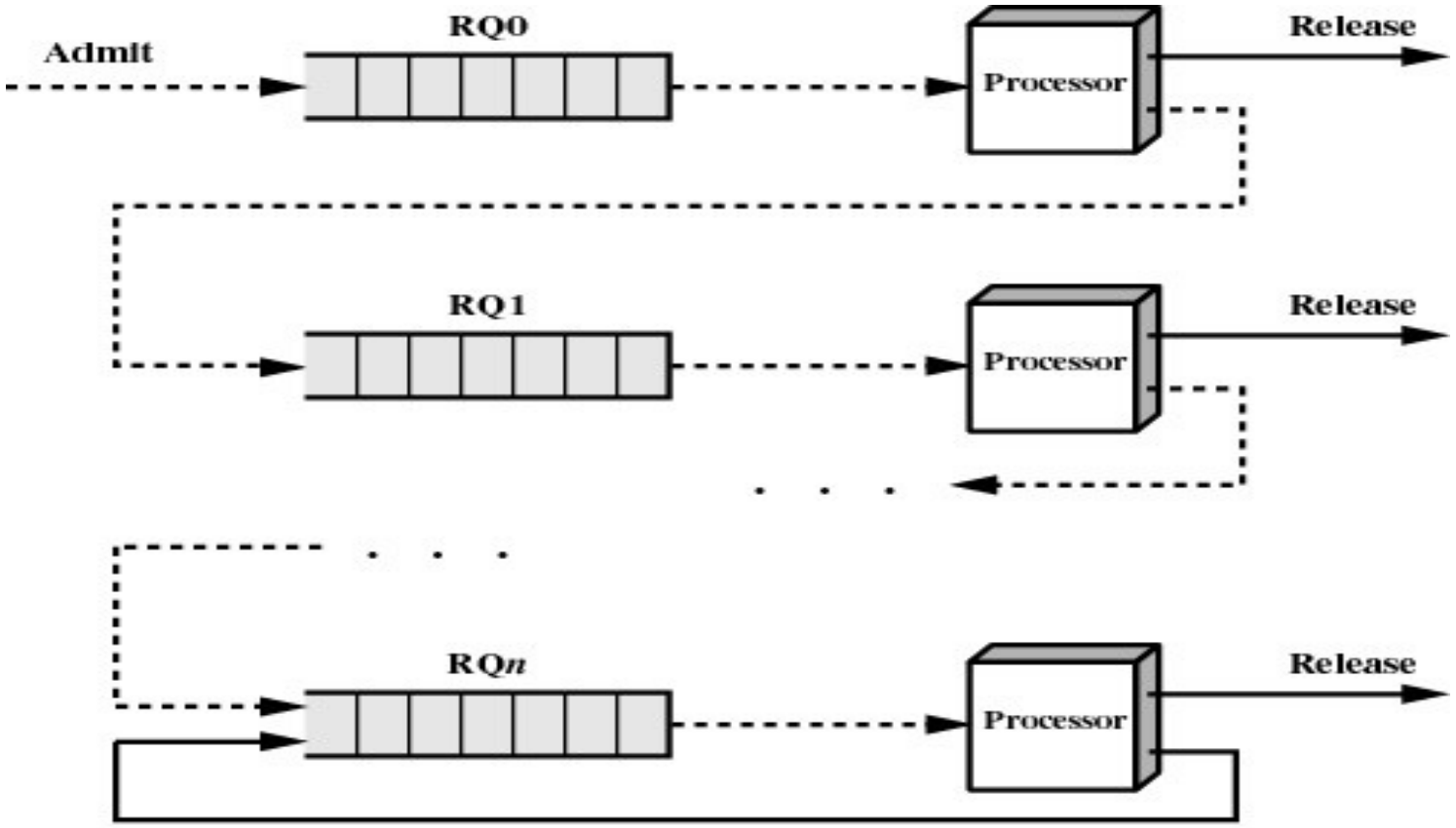
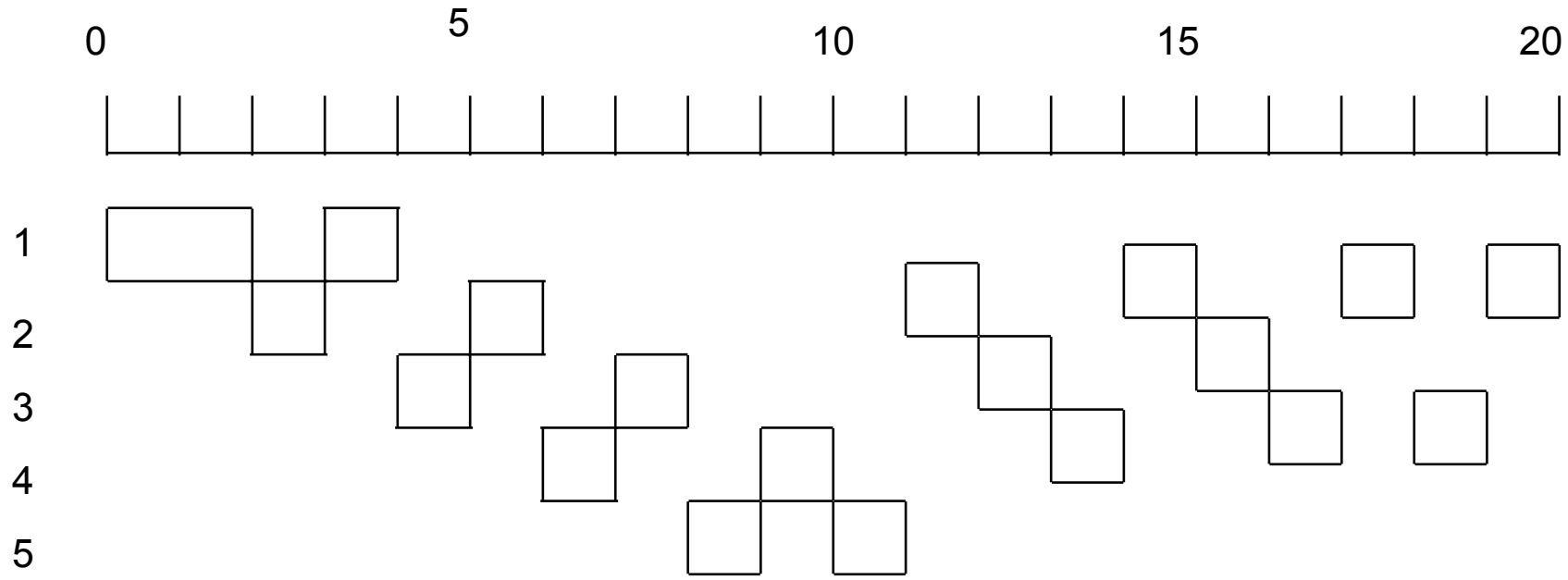


Figure 9.10 Feedback Scheduling

Feedback (FB，反馈调度法)



Feedback (FB，反馈调度法)

算法分析：

1. 有利于交互型作业 （常为短作业）
2. 有利于短批处理作业
3. 存在的问题：

- 当不断有新进程到来，则长进程可能饥饿。

系统实例分析

UNIX 、 LINUX 系统调度实例