2021219113 2021213595 沈尉林

分别考虑每种情况下平均完成时间最小的情况：

1. FCFS（先来先服务）：假设任务的到达顺序为A、B、C、D、E，估计的运行时间分别为3、4、2、5、6。在FCFS中，任务按照它们到达的顺序执行。因此，平均完成时间为(3+7+9+14+20)/5 = 10.6 时间单位。
2. SJF（最短作业优先）：假设任务的到达顺序为A、B、C、D、E，估计的运行时间分别为2、3、4、5、6。在SJF中，任务按照它们的估计运行时间进行排序，然后执行最短的任务。因此，平均完成时间为(2+5+9+14+20)/5 = 10 时间单位。
3. RR（时间片轮转）：假设任务的到达顺序为A、B、C、D、E，估计的运行时间分别为4、4、4、4、4，而时间片大小为2。在RR中，任务按照时间片轮转的方式执行，每个任务执行2个时间单位，然后切换到下一个任务。因此，平均完成时间取决于时间片大小，如果时间片足够小，平均完成时间可以接近SJF。
4. EDF（最早截止期限优先）：假设任务的到达顺序为A、B、C、D、E，估计的运行时间分别为2、3、4、5、6，截止期限分别为2、5、9、14、20。在EDF中，任务按照它们的截止期限进行排序，然后执行最早截止的任务。因此，平均完成时间为(2+5+9+14+20)/5 = 10 时间单位。

CFS（Completely Fair Scheduler）调度算法

CFS的核心思想是公平分配CPU时间，以便所有运行的进程（任务）在相同的时间段内都获得相等的CPU时间分配。它使用红黑树数据结构来跟踪任务的运行时间和优先级，以便在每个时间片内选择最需要CPU时间的任务。CFS试图避免不公平的CPU时间分配，确保所有任务都能够充分利用CPU资源。

CFS相对于一些传统的调度算法的优势包括：

1. **公平性**：CFS致力于确保每个任务都能够公平分享CPU时间，避免了饥饿和不公平的情况。
2. **动态优先级**：CFS不依赖于静态的优先级值，而是根据任务的运行情况动态分配CPU时间，这有助于更好地响应不同任务的需求。
3. **高度可扩展**：CFS的红黑树数据结构使其能够有效地处理大量任务，因此适用于多核系统和大规模服务器。