高级操作系统

5. 分布式资源管理

熊焰，yxiong@ustc.edu.cn
黄文超, huangwc@ustc.edu.cn
http://staff.ustc.edu.cn/~huangwc/advancedos

参考书目：《分布式计算机系统—孙忠秀》

书籍下载链接
http://www.sslibrary.com

本章内容

1. 资源管理的基本概念
2. 集中分布资源管理
3. 完全分布资源管理

1. 基本概念

- 分布式OS和单机（集中式）OS的主要区别
  - **资源管理**
  - 进程通信
  - 系统结构

分布式OS和单机（集中式）OS的主要区别：
资源管理
进程通信
系统结构
资源管理是操作系统的一项主要任务：

1. 基本概念

- 单机OS的资源管理
  - 采用**一类资源**由**一个资源管理者**来管的集中式管理方式。
- 注：分布式计算机系统中，这种方法会使性能很差，如
  - 开销大
  - 鲁棒性差

单机OS的资源管理：采用一类资源由一个资源管理者来管的集中式管理方式。例如，所有内存都由存储管理负责分配和释放；所有行打机由打印机管理负责打印等等。
在分布式计算机系统中，由于系统资源是分布在各台计算机上的，若一类资源归一个管理者来管里会使性能很差。假如，系统中各台计算机的存储资源由位于某台计算机上的资源管理者来管，那么，不论谁申请存储资源，即使申请的是自己计算机上的资源，都必须发信给存储管理，这就大大增加了系统开销。如果存储管理所在那台计算机坏了，系统便会瘫痪。由此可见，分布式操作系统采用集中式方式来管理资源，不仅开销大，而且鲁棒性差。

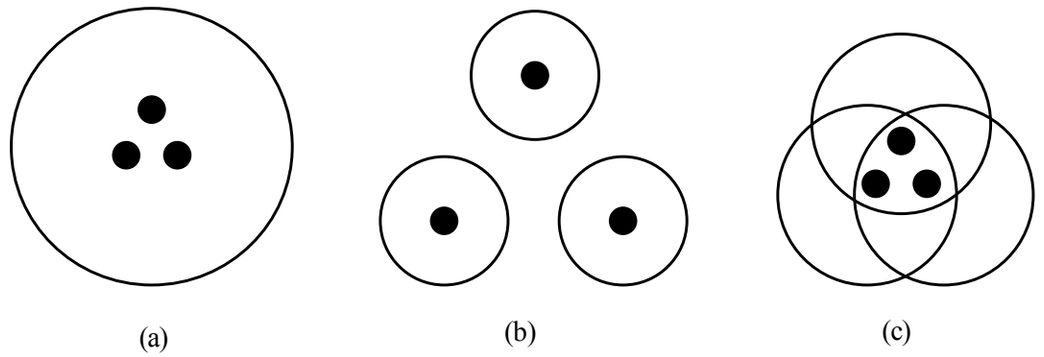
1. 基本概念

- 分布式OS的管理——**一类**资源，多个管理者
  - 集中分布管理
    - **每个**具体资源只存在唯一的一个管理者对其负责
  - 完全分布式管理方式
    - **一个**资源由多个管理者共同管理

分布式OS的管理：分布式操作系统采用一类资源多个管理者的方式。例如，系统有若干个位于不同计算机上的文件管理。它们可以共同管理，也可以分别管理系统中的文件。
两种分布式管理方式：
集中分布管理：一类资源由多个管理者管理，但每个具体资源只存在唯一的一个管理者对其负责。比如上述文件管理，尽管系统有多个文件管理，但每个文件只依属于一个文件管理。
换言之，在集中分布式管理下，使用某个文件必须也仅需通过与其依属的某个文件管理。
完全分布式管理方式：一类资源由多个管理者管理，但一个资源由多个管理者共同管理。假如一份文件有若干分文件副本，这些副本分别受管于不同的文件管理。为了保证各副本的一致性，当一份副本正在被修改时，其它各副本应被禁止使用。因此，当一个文件管理接到使用文件的申请时，它只在和管理该文件其它副本的管理者协商之后，才能决定是否让申请者 使用文件。在这种情况下，一个具有多副本的文件资源是由多个文件管理共同管理的。

分布式和集中式管理方式的主要区别：对同类资源采用多个管理者还是一个管理者。
集中分布式和完全分布式管理方式的区别：前者对所管资源拥有完全控制权，一类资源中的每一个资源仅受控于一个资源管理者；而后者对所管资源仅有部分控制权，不仅一类资源存在多个管理者，而且该类中每个资源都由多个管理者共同控制。

1. 基本概念



（a）集中管理方式，（b）集中分布管理方式，（c）完全分布管理方式

图给出了这三种不同的资源管理方式的示意。图中，一点表示一个资源；圈表示相应的资源管理，它管理圈内的资源。采用集中管理方式时，一类资源只有一个管理者，它控制该类全部资源。采用集中分布管理方式时，一类资源由多个管理者来管，但每一个资源只受控于一个管理者。采用完全分布管理方式时，不仅一类资源存在多个管理者，而且该类中每个资源都由多个管理者对其控制。

1. 基本概念

- 从两种管理方式的角度，资源可分为两大类
  - 和处理机紧密相连的资源～集中分布管理
    - 如存储单元、显示器、硬盘以及与计算机连接的打印机等
  - 和处理机关系不甚紧密的资源～完全分布管理
    - 多副本文件、多台处理机相连的打印机

集中分布管理方式比较容易实现，因为每个管理者管理资源的方式和集中管理方式基本上一样。完全分布管理方式实现起来比较复杂。为了保证系统的鲁棒性，对某些资源（例如共享文件）必须采用完全分布管理方式。

从两种管理方式的角度来考虑系统资源的划分，资源可分为两大类：

和处理机紧密相连的资源：如存储单元、显示器、硬盘以及与计算机连接的打印机等，当与它们紧密相连的计算机失效时，这些资源也就失效了。对于这种类型的资源往往采用集中分布管理方式。资源的管理者就放在被管理资源所连的那台处理机上。

和处理机关系不甚紧密的资源：如多副本文件。与多台处理机相连的打印机等，当一台处理机失效时，通过别的处理机仍可使用这类资源。对于这种类型的资源，往往采用完全分布管理方式。

1. 基本概念

- 集中资源申请
  - 无死锁、无饿死
- 集中分布资源申请
  - 无死锁、有饿死
- 完全分布式管理
  - 有死锁、有饿死

一般来说，一个分布式操作系统往往兼有两种管理方式。

集中和分布式资源申请过程的区别：

集中资源申请：资源的申请者总是向唯一的一个资源管理者提出申请，因此，申请者可以按一个确定的次序排队等候。对于这种情形，只要不发生死锁，并且任何资源占有者都能在有限长的时间内释放所占用的资源，那么，任何 申请者必定能在有限长的时间内获得资源。

集中分布资源申请：一个申请者先向某个管理者提出申请。当申请者得知暂时不能获得所需资源后，应向另一个管理者提出申请。因而会产生饿死现象
饿死（或饥饿）现象：申请者A向资源管理者R1申请资源，R1的资源不空，A转向资源管理者R2，此时，R1的资源刚被释放，且正逢另一个申请者B向R1申请，因而，B获得资源。A向R2申请资源又被拒绝，而当A第二次向R1申请资源时，R2资源恰好空了，但又被另一个申请者C占用了，R1仍不能满足A的申请，因为它的资源已被B占用；如此下去，B和C不断地从R1和R2处获得资源、使用资源、归还资源，而A交替地向R1和R2提出申请却永远得不到资源。这种现象和死锁不同。当发生死锁时，一定有一个资源被无限期地占用而得不到释放。而现在的情形是，每个资源占有者都在有限长的时间内释放它所占有的资源，但仍然存在着申请者得不到资源。我们把这种现象称为“饿死”。

在完全分布式管理方式下资源的分配是通过几个管理者协商而定的。如果协商的原则规定的不好，就可能产生“饿死”现象，即某个申请者经过每次协商后都得不到所要的资源。因此，设计分布式操作系统时，不仅要考虑如何防止“死锁”，还要考虑如何避免“饿死”。进一步讲，分配资源的算法应能满足如下条件：任何资源的占用者总能在有限长的时间内释放所占用的资源，并且任何资源申请者总能在有限长的时间内获得资源。

2. 集中分布资源管理

- 集中分布管理方式和集中管理方式的区别在于
  - 前者应具有向其它资源管理者提交申请和接受其它资源管理者转来申请的功能
- 所以，必须制定一个**资源搜索算法**，使得资源管理者按此算法帮助用户找到所需资源

采用集中分布管理方式时，每个资源均由唯一的一个管理者管理。每个资源管理者所进行的分配、释放工作，类似于集中管理方式。但是，当一个资源管理者不能满足一个申请者的请求时，它应当帮助用户去向其它资源管理者申请资源。这样用户申请资源的过程类似在单机操作系统上一样，只要向本机的资源管理者提出申请，他无须知道系统中有多少个资源管理者，也无须知道资源的分布情况。集中分布管理方式和集中管理方式的区别在于，前者应具有向其它资源管理者提交申请和接受其它资源管理者转来申请的功能。

既然系统有多个资源管理者，它们分布在不同的计算机上，资源管理就必须根据一定的策略和规则，依次询问各个资源管理。换句话说，必须制定一个资源搜索算法，使得资源管理者按此算法帮助用户找到所需资源。

2. 集中分布资源管理

- 资源搜索算法的**目标**
  - 避免饿死
  - 高效率地利用资源
  - 资源使用均衡
  - 算法开销小
  - 算法具有鲁棒性

资源搜索算法满足下列条件：

避免饿死：只要每个资源申请者都能在有限长的时间内获得所需资源，则按算法搜索一定能获得资源。

高效率地利用资源：使用资源不会出现舍近求远的现象。

资源使用均衡：不应使某些资源使用过于频繁，而另一些资源使用过于清闲。

算法开销小：执行算法时通信量少。

算法具有鲁棒性：在节点失效时算法仍有效。

2. 集中分布资源管理

- 前提假设
  - 信件的传递满足“先发先到”的条件
    - 即一节点向另一节点先后发送了两封信，先发出的信一定先收到。
  - 节点未失效时，信件一定能无误地被接收
    - 失效的节点不再被外界感知，即它即不能发出信件，也不对任何信件作出反映。

资源搜索算法的两点假设：

信件的传递满足“先发先到”的条件，即一节点向另一节点先后发送了两封信，先发出的信一定先收到。

节点未失效时，信件一定能无误地被接收，失效的节点不再被外界感知，即它即不能发出信件，也不对任何信件作出反映。

2. 集中分布资源管理

- 投标算法
- 回声算法
- 由近及远算法

2. 集中分布资源管理
投标算法

- 1. 资源管理者欲向它机资源管理者申请资源时，首先**广播招标消息**，向网络中位于其它结点的每个资源管理者发招标消息。
- 2. 当一个资源管理者接到招标消息时，如果该结点上有所需资源，则根据一定的策略**计算出”标数”**，然后**发一个投标消息**给申请者，否则回一个拒绝消息。
- 3. 当申请者收到所有回答消息后，根据一定策略**选出一个投标者**，并向**它发一个申请消息**。
- 4. 接到申请消息后，将申请者的名字登记入册，并在可以分配资源时**发消息**通知申请者。
- 5. 当资源使用完毕后，向分配资源的资源管理者归还资源。

2. 集中分布资源管理
投标算法

- 标数规定：b=w1*a+w2*d
  - a为等待申请者的个数，
  - d为投标者与招标者间的距离；
  - w1和w2为两个常数。
- 采用这种投标策略即考虑了资源的使用的**均衡性**，又兼顾了资源使用的**有效性**。

2. 集中分布资源管理
投标算法

- **但是**，上述算法**没有考虑节点失效的情况**，增加下两条将使算法具有鲁棒性：
  - 5. 若发申请后很久未获得资源，则向中标者发一询问信：“你还在吗？”。若中标者未失效就立即予以肯定答复。若发询问信后未见回答，则重新广播招标信件。
  - 3. 改为“… 接到所有回信后，或等回信等了较长时间后，…”。
- 按招标算法搜索资源**不会出现饿死现象**。

因为只要系统中有所申请的资源就必定有一个中标者。只要每个占有资源者在有限长的时间内归还所有资源，申请者总能从中标者那里获得资源。

2. 集中分布资源管理
投标算法

- 效率分析
  - 在没有节点失效时，从广播招标信到接到获得资源通知，一共发了M封信

$$M=2(n-1)+2=2n$$

  - 此处n为网络中的节点总数。

在没有节点失效时，从广播招标信到接到获得资源通知，一共发了M封信:

M=2(n-1)+2=2n

此处n为网络中的节点总数。

投标算法是一个简单而实用的搜索算法，美国加州大学欧文分校设计的DCS分布式计算机系统便采用了投标算法。

2. 集中分布资源管理
投标算法——环形结构系统

- 1. 需求资源者向其邻居节点发一封招标信。
- 2. 接到招标信后，若本节点上无此类资源，则将招标信沿环传向下一邻居节点，否则
  - (1)若信中未附投标，则将本节点的投标附上，将信传给下一邻点；
  - (2)若信中已附有投标，则将本节点的投标和它比较，优选一个附在信中传向下一个邻点。
- …（后面类似）

3. 接到自己发出的招标信后，从信中所附投标可知中标的资源管理者是谁。
4. 向中标的资源管理者发一封申请信。
5. 中标者接到申请信后将申请者排入申请队列，并在可以使用资源时向它发出通知。
5. 使用资源完毕后，通知分配资源者收回资源。

2. 集中分布资源管理
回声算法

- 1.资源申请者向它的**每一个邻结点发探查消息**，消息中附上对资源的需求。
- 2.若接探查消息的结点是**第一次接到**这样的探查消息
  - 它就把传来探查消息的邻结点定义为它的对该探查而言的**上邻结点**，而把**其余**的邻结点定义为它的**下邻结点**。
- 若接探查消息的结点**不是第一次**接到这样的探查消息，
  - 它就向传来探查消息的邻结点发**一回声消息**，消息中**参数值为0**。

回声算法：

回声算法是用来获得全局知识的一种算法。它也可用于搜索资源，用于搜索资源的回声算法由以下规则来定义：

2. 集中分布资源管理
回声算法

- 3.接上邻结点传来的探查消息后
  - 若有下邻结点
    - 则将探查消息复制后**分发给各下邻结点**
  - 否则
    - 向**上邻结点发一回声消息**消息中参数S(称资源参数)取值如下：
      - S=0　　当结点不具备所需资源时
      - S=w*a+1　当有a 个申请者在等待资源时
        - 式中w是一个常数。

2. 集中分布资源管理
回声算法

- 4.当一个结点**接到**它的**所有下邻**结点发来的**回声**消息后，它就向它的**上邻结点发一回声消息**，消息中附上参数**S**及**与之对应的结点编号**。参数S的取值如下：
  - S=0　　若Sr=0 且所有回声消息中所附参数均为0
  - S=min(Sr1+1,…,Sre+1,Sr)
    - 式中,Sr为本结点的资源参数；Sr1,…,Sre 为所有回声中所附的非零资源参数。
      - 若S值被选为Sre+1，那么回声消息中所附结点编号就是附有资源参数Sre的回声中**所附的结点编号**。
      - 若S值被选为Sr，则回声消息中所附结点编号就是**本结点的编号**。

2. 集中分布资源管理
回声算法

- 5.申请者获得所有邻结点发来的回声消息后，将按上一条规则选定 S的方法**选中一个资源提供者**，然后，向它**发申请消息**。
- 6.当一个结点接到申请消息后，就把申请消息登记下，并在可能时将资源**分配**给它。
- 7.使用完毕后通知资源分配者**释放**。

2. 集中分布资源管理
回声算法

- 为了使算法具有**鲁棒性**还需要补充如下两条规则：
  - 8.如果某个下邻节点**很久没有回声**，则发信询问：“你还在吗？”，
    - 如果得到肯定回答则继续等待；
    - 否则，就假定它的回声信已经收到，信中资源参数为0。
  - 9.如果发出申请信后**很久未得到资源**，则询问资源提供者“你还在吗？”
    - 如果得到肯定回答，则继续等待；
    - 否则，重发探查信。

为了使算法具有鲁棒性还需要补充如下两条规则：
8.如果某个下邻节点很久没有回声，则发信询问：“你还在吗？”，如果得到肯定回答，则继续等待；否则，就假定它的回声信已经收到，信中资源参数为0。
9.如果发出申请信后很久未得到资源，则询问资源提供者“你还在吗？”如果得到肯定回答，则继续等待；否则，重发探查信。

2. 集中分布资源管理
回声算法

- 按回声算法搜索资源**不会产生饿死现象**
- 但是，它的**通信量**比投标算法要高。
- 优化思路：对于一个**节点很多的系统**往往**没有必要**去搜索所有的节点。我们希望，只要**找到满意的资源**就**停止搜索**。

2. 集中分布资源管理
回声算法

- 改进方案：
  - （2）若接探查信的节点**不具有所需的资源并且没有收到过同样的探查信**，则将探查信**转发**给它的所有的下邻节点。
  - （3）若接探查信的节点**未收到过同样的探查信**，并且它**具有所需资源**，则向来信者**发一封回信**。
  - （4）当一个节点**接到满意的回声信**或**所有下邻节点的回声信后**，即**向上邻节点发回声信**。信中所附的资源参数仍按原规则确定。
  - （5）当**申请者**收到**满意的回声信或全部邻点的回声信后**，就选择一个资源提供者，向它申请资源。

采用改进的回声算法搜索资源时，不仅不必搜索所有节点，而且不具有资源的节点不发回声信。这样，就大大减少了通信量

2. 集中分布资源管理
由近及远算法

- 1.资源申请者向它的**某个邻结点发一个搜索消息**，信中附上对资源的需求及**参数 P**，其值为**申请者编号**。
- 2.接搜索消息后，将发来消息的结点编号和信中参数 p登记下来，
  - 前者**定义为它的上邻结点**，后者定义为它的**前结点**。
  - **如果**接搜索消息的结点**具有**消息中所要求的**资源**，那么，它就向它的**上邻**结点**发一个成功消息**，并将自己的编号附上；
  - **否则**它先发一个消息给它的前结点**告知自己是它的后结点**。然后，发消息给上邻结点，请继续搜索，消息中带上**参数 p**，其值**为自己的编号**。

该算法让资源申请者由近及远地搜索，直到搜索到具有所要资源的结点为止。

2. 集中分布资源管理
由近及远算法

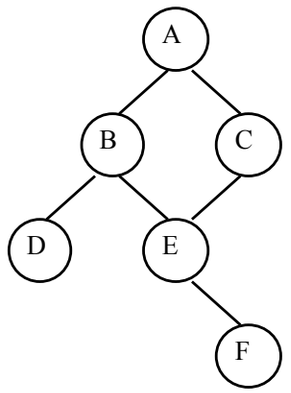
- 3. 接继续搜索消息后
  - 如果**还有未被搜索**的下邻结点，那么，就发搜索消息给它，消息中附上的参数 p是从继续搜索消息中取得的。
  - 如果所有下邻结点都已搜索过，但它**有后结点**，则把继续搜索消息转给它的后结点。
  - 如果既没有未被搜索的下邻结点，又没有后结点，则说明**全部结点已被搜索过**，这时它将向上邻结点发一个失败消息。

2. 集中分布资源管理
由近及远算法

- 4. 接成功消息或失败消息后
  - 若接消息者非申请者，则将消息转发给它的上邻结点，否则搜索到此结束。
  - 申请者或获得最近能提供所要资源的结点地址或被告之系统中没有这样的资源。
- 5. 如果一个已被搜索过的结点**又收到搜索消息**，则将原消息**退回**，发搜索消息的结点**就认为该下邻结点不存在**。

2. 集中分布资源管理
由近及远算法

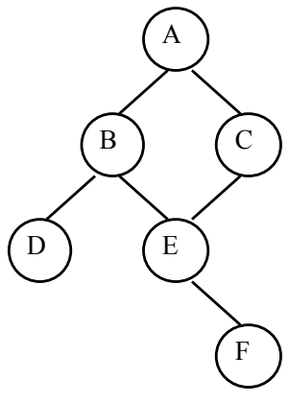
- （1）A向B发搜索信；（规则1）
- （2）B向A发信告知A的后节点是B；（规则2）
- （3）B向A发信请继续搜索；（规则2）
- （4）A向C发搜索信；（规则3）
- （5）C向B发信告知B的后节点是C；（规则2）
- （6）C向A发信请继续搜索；（规则2）
- （7）A将继续搜索信转给它的后节点B；(规则3)
- （8）B向D发搜索信；（规则3）
- （9）D向C发信告知C的后节点是D；（规则2）
- （10）D向B发信请继续搜索；（规则2）
- （11）B向E发搜索信；（规则3）



举例说明由近及远算法的工作情况。假定在图5－2所示的系统中，节点A时申请者，只有节点F具有A要的资源，按由近及远算法，搜索过程如下：

2. 集中分布资源管理
由近及远算法

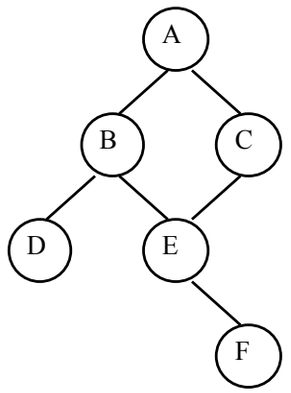
- （12）E向D发信告知D的后节点是E；（规则3）
- （13）E向B发信请继续搜索；（规则2）
- （14）B将继续搜索信转给它的后节点C；(规则3)
- （15）C向E发搜索信；（规则3）
- （16）E将搜索信退还C；（规则5）
- （17）C将继续搜索信转给它的后节点D；(规则3)
- （18）D将继续搜索信转给它的后节点E；(规则3)
- （19）E向F发搜索信；（规则3）
- （20）F向E发成功信；（规则2）
- （21）E向B转发成功信；（规则4）
- （22）B向A转发成功信；（规则4）
- （23）A收到来自F的成功信。



举例说明由近及远算法的工作情况。假定在图5－2所示的系统中，节点A时申请者，只有节点F具有A要的资源，按由近及远算法，搜索过程如下：

2. 集中分布资源管理
由近及远算法

- 如果F也没有A所要的资源，则（19）步以后的过程将改为：
  - （20）F向E发信告知E的后节点F；（规则2）
  - （21）F向E发信请继续搜索；（规则2）
  - （22）E将继续搜索信转给它的后节点F；(规则3)
  - （23）F即无资源又无后节点，因此,F发失败信息给它的上邻节点E；（规则3）
  - （24）E向B转发失败信；（规则4）
  - （25）B向A转发失败信；（规则4）
  - （26）A收到失败信，搜索失败。



举例说明由近及远算法的工作情况。假定在图5－2所示的系统中，节点A时申请者，只有节点F具有A要的资源，按由近及远算法，搜索过程如下：

2. 集中分布资源管理
由近及远算法

- 采用由近及远算法搜索资源**不会产生饿死现象**。
- 它比前两种算法通信量大的多。
  - 每个节点至少收三封信，发一封信
- 但当系统中有**较多的节点拥有资源时**，采用这种算法**很快就能获得资源**。
- 因此，对于**“稀有”资源**，投标算法或回声算法比较合适，而对于**普遍资源**，由近及远算法可能更好些。

2. 完全分布资源管理

- 设计目标
  - 应保证**每个资源**在**任何时刻最多**被**一个进程所占有**，即保证资源分配的**互斥性**。
  - 不应产生**饿死**现象。
- 算法
  - 里卡特算法
  - 令牌算法

采用完全分布式管理方式时，每个资源由位于不同节点上的资源管理共同来管。每个资源管理在决定分配它所管理的资源前，必须和其他资源管理者协商。因此,采用这种管理,需要设计一个算法，各资源管理按照该算法给出的原则共同协商资源的分配。这个算法应满足以下条件：

2. 完全分布资源管理
里卡特算法

- （1）申请资源者向网络中所有进程**广播申请信**，信上加盖申请时刻的**时间戳**，一旦**收到所有进程的回答**，就可以**获取资源**。
- （2）进程**Pr接到**一封来自进程**Ps**的申请信时，设信上的时间戳为**Ts**，
  - 如果Pr**既不是**资源的**申请者**又**不是**资源的**占用者**，则立即**给Ps以回答**；
  - **否则**，**仅当Pr不是**资源的**占有者**（它必定是资源的申请者），并且满足条件:

    $$Ts<Tr \text{ 或 } Ts=Tr且s<r$$

    时才予以回答。

里卡特（G.Ricart）等人于1981年提出了一个成为最佳算法的分布式同步算法。这个算法实际上是一个改进的时间戳算法。

上述条件中，Tr为Pr申请信中的时间戳，s和r分别为进程Ps和Pr的编号。

2. 完全分布资源管理
里卡特算法

- （3）占用资源的进程在**释放资源时**，对那些曾经接到过它们的申请信但未予回答的进程，**补送回答**。

- （4）（补充）各进程都定义了一个逻辑时钟，每当发生一个诸如发信、收信等事件时，它的值将按**逻辑时钟的定义增大**。

- 该算法保证资源的**互斥分配**，并且**不会产生饿死现象**。

2. 完全分布资源管理
里卡特算法

- 为了使算法具有**鲁棒性**，需要考虑节点失效的情况，为此增加两条规则：

- （5）发出申请信后，如果**某进程久未回答**，就向**他发一封探查信**。如果发探查信后仍不见回答，那么这个被探查的节点必定已经失败，因此就不再等它的回答了。

- （6）**若接到探查信**，立即**发一封确认信给探查者**，以确认自己在正常工作。

2. 完全分布资源管理
令牌算法

- （1）每个进程有一张记录**各进程申请资源状态**的**表格**。
- （2）**初始化时**，有且只有**一个进程持有令牌**。
- （3）只有**令牌持有者**才能获得**资源**。
- （4）申请资源时，如果该进程不持有令牌，则向其它进程**广播申请信**，信中附上当时的**时间戳**（其值大于表中所记录的所有其它进程发来申请信中的时间戳的值）。

令牌算法是一种通信量更小的分布式同步算法。

（1）当它发申请信时，或者接到其它进程发来的申请信时，就记录下该进程已经处于申请资源状态，并将信中所附的时间戳记录下来。
注意：接到某个进程发来的第二封申请信时，第一封信的时间戳就不再保留了。

2. 完全分布资源管理
令牌算法

- （5）如果**令牌持有者**不是申请者并且它**不在使用资源**，则当表中记录有申请者时，就**选择一个具有最小时间戳**的申请者，将令牌**传给它**，并附上自己表中保存的所有进程使用资源的状况（是否在申请，以及最大的时间戳）。

- （6）**收到令牌的进程**，根据令牌中附有的各进程状况，**对自己表**中各个进程的状况，以时间戳最大的值为标准**进行修改**，从而成为令牌持有者。

- （7）**每次使用互斥资源后**，将**增加表中自己**所对应栏的**时间戳**的值并将自己的状态改为**非申请资源状态**，然后**转规则（5）**。

2. 完全分布资源管理
令牌算法

- **该算法不具有鲁棒性**，特别是当**令牌丢失后**（令牌持有者失效），必须有一个算法来决定谁能成为令牌持有者。为了使令牌算法具有鲁棒性，需要**增进如下几条规则**：

  - （8）当申请信发出后**很久未获得令牌**时，向其它各进程**广播**探查信。如果其它各进程**没有回信反对**，那它就成为令牌持有者。**否则继续等待**。

  - （9）**接探查信后**，**如果**接信者是令牌持有者，或者是申请者，并且自己申请时的时间戳小于发探查信者的时间戳（相等时，则比较进程编号），则立即发一封**反对信**。

  - （10）令牌持有者**传送令牌时**，如果**发现接收者失效**，则**重新选择申请者**，以传送令牌。