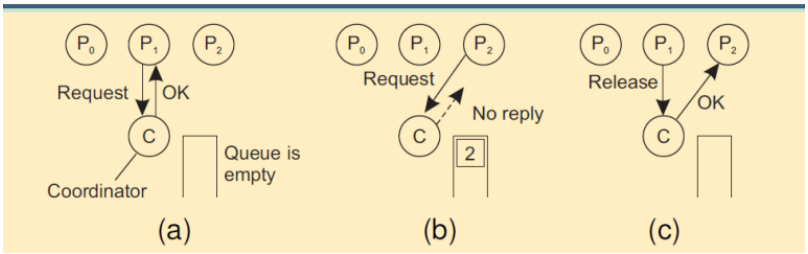
互斥：

- 互斥：
  - 背景：分布式系统多个进程需要互斥访问资源
  - 基于许可的方法：
    - 基本思想：从管理资源的进程获得许可
    - 基于许可的集中式方法：
      - 过程：



        - P1向C申请资源许可，等待队列为空且无进程申请许可，C给许可P1
        - P2向C申请许可，但是C在P1在使用的情况下不会给许可，P2进入等待队列
        - P1释放资源，C得得知P1释放了，通知等待队列中的P2 ok
      - 问题：协作者负载超级高性能低，而且一崩溃，没有人可以用资源
    - 非集中式算法：
      - 假设资源有N个副本（虚拟的），每个副本有一个协作者。协作者只有一次投票权，当进程获得m>N/2个协作者的投票，就可以访问资源。协作者可以立即响应请求（N为总节点数）

        所有节点都是协作者
      - 前提：协作者崩溃会马上回复，但是忘记自己发出去过的许可
      - 存在问题：可能所有申请资源的进程都拿不到足够的赞成票P1,P2,P3在5个协作者情况下申请资源（需要拿到3票及以上的赞成票），P1:2, P2:2,P3:1，没有一个进程拿得到

        解决：进程释放自己的票，不停的重新申请
      - 目的：减少单点故障，同时提高性能
      - 单个协作者失效概率：

$$\text{et } p = \Delta t / T \text{ b}$$

      - m个协作者同时有k个协作者失效的概率：

$$\mathbb{P}[k] = \binom{m}{k} p^k (1-p)^{m-k}$$

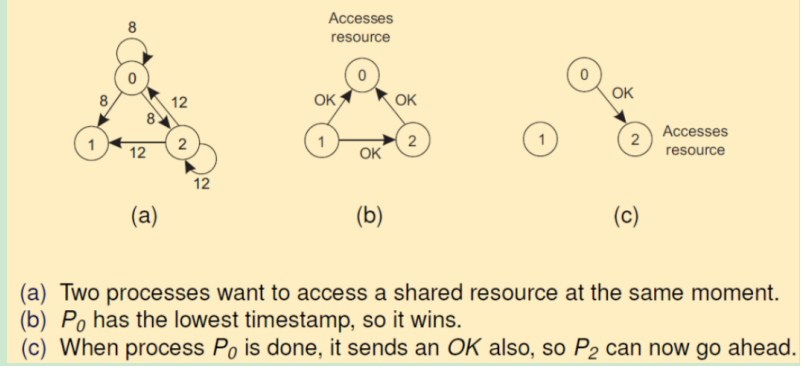
- 算法失效：正确的协作者的数量不超过总节点（N）的1/2，也就是失效的节点数f>=m(总协作者数)-N/2

  - f coordinators reset ⇒ correctness is violated when there is only a minority of nonfaulty coordinators: when $m - f \leq N/2$, or, $f \geq m - N/2$.

- 算法失效概率：

  - The probability of a violation is $\sum_{k=m-N/2}^{N} \mathbb{P}[k]$.

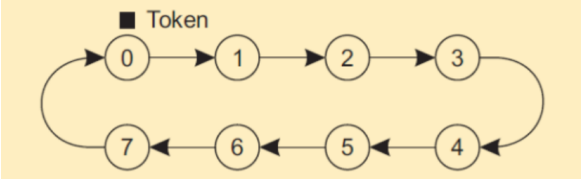
- 分布式算法：
  - 背景：避免单点故障
  - Ricart&Agrawala算法：
    - 要求系统中的事件完全排序，用时间戳物理时钟
    - 进程要访问共享资源时，构造一个消息，包括资源名，进程号和当前逻辑事件，发送给所有其他进程
    - 接收到消息的决策动作：
      - 接收进程没有也不想访问资源：返回ok
      - 接收者已经在访问资源了：不ok，将请求放在队列中
      - 接收者也想访问资源：比较和自己发出去的时间戳，事件戳早的获胜，如果一样就比较pid小的获胜，自己赢了就不ok，请求放队列，没赢就返回ok
    - 例子：在收到其他所有进程的ok才能访问进程

      (a) Two processes want to access a shared resource at the same moment.
      (b) P_0 has the lowest timestamp, so it wins.
      (c) When process P_0 is done, it sends an *OK* also, so P_2 can now go ahead.

    - 问题：单点失效变成多点失效，只要所有进程有一个失效就完全失效，单点失效的概率1-p，多点失效的概率$1-(1-p)^2$；进程维护开销大

- 基于令牌的方法：
  - 基本思想：一个资源一个令牌在进程间传递，拥有令牌的进程可以访问临界资源或将令牌传递给其他进程
  - 问题：令牌传丢了，拥有令牌的进程死了
  - 本质：将进程组织成逻辑环，令牌在进程之间传递，拥有令牌的进程允许进入临界区

- 互斥算法对比：每个进入+请求的总消息量

Algorithm	Messages per entry/exit	Delay before entry (in message times)
Centralized	3	2
Distributed	$2 \cdot (N-1)$	$2 \cdot (N-1)$
Token ring	$1, \ldots, \infty$	$0, \ldots, N-1$
Decentralized	$2 \cdot m \cdot k + m, k = 1, 2, \ldots$	$2 \cdot m \cdot k$

  - 集中式许可：3次（申请一来一回2+退出1），2要等申请消息传递和响应传回
  - 分布式许可：进入都需要所有进程总共发送2*(N-1), 申请许可的时间一来一回也是2*（N-1），退出不发送消息（那个占有资源时不ok，但是释放了就ok了）
  - 令牌环：消息传递1到无穷（传丢了），延迟时间0..N-1(转了一圈，令牌才到)
  - 非集中式许可：m个协作者，通知m个协作者投票，协作者投票，一个来回一共2m个消息，但是又可能要重试k次。为什么要再加m，退出时只用发m次