高级操作系统期末总复习

第一章

1.什么是分布式系统?

分布式系统是若干独立计算机的集合，它们对于用户来说就像一个系统。

2.分布式系统中透明性的种类、定义

透明性定义：分布式系统的重要目标之一是透明性，即将它的进程和资源实际上分布在多台计算机上这一事实隐藏起来。

透明性种类：

透明性	描述
访问	隐藏数据表示形式以及访问方式的不同
位置	隐藏数据所在位置
迁移	隐藏资源是否已移动到另一个位置
重定位	隐藏资源是否在使用中已移动到另一个位置
复制	隐藏资源是否已经被复制
并发	隐藏资源是否由若干相互竞争的用户共享
故障	隐藏资源的故障和恢复
持久性	隐藏资源位于内存里或在磁盘上

访位迁重复发障九

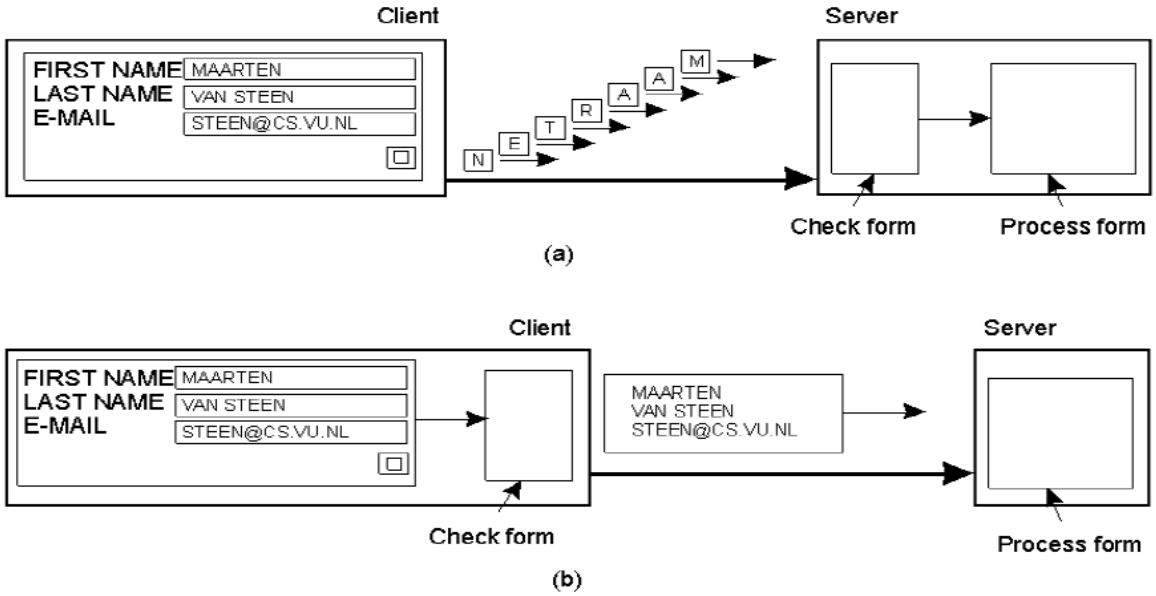
3.分布式系统中的拓展技术有哪些?

规模上拓展、地域上拓展、管理上拓展。

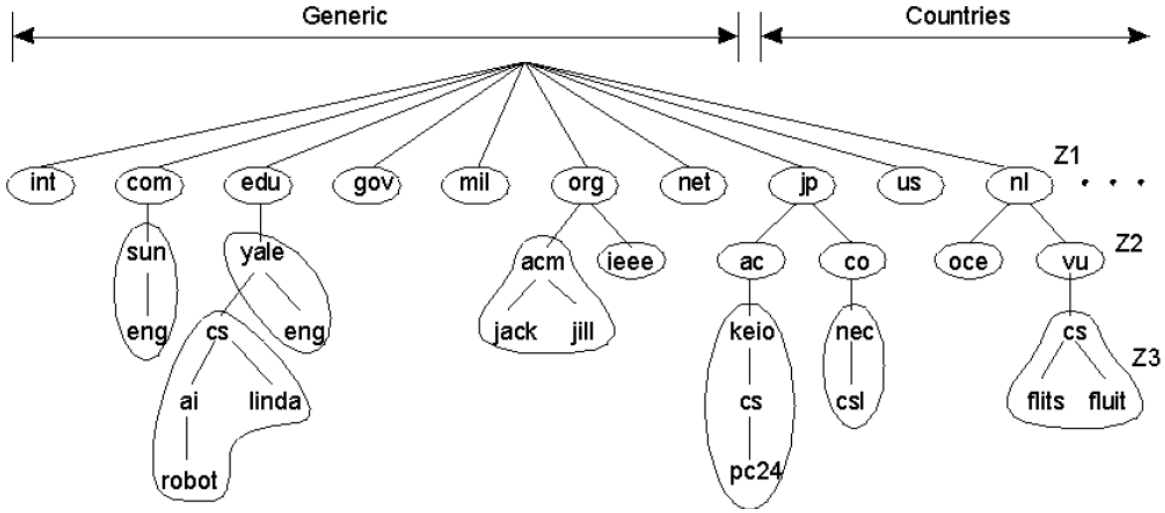
分布式系统中的可拓展性问题表现为服务器和网络能力有限所造成的性能问题，因此基本上只有三种拓展技术：隐藏通信等待时间、分布技术、复制技术。

隐藏通信等待时间：异步通信、减少通信时间。

减少通信时间——客户端检查表单



分布技术：DNS服务、WWW服务。



复制技术：多拷贝。

复制：增加可用性，有助于负载均衡。

缓存：在访问资源的客户周围制作资源备份。

一致性问题。

第二章 体系结构

1.客户端-服务器模型

集中式体系结构——客户端-服务器模型

服务器：实现某个特定服务的进程；

客户：向服务器请求服务的进程；

客户端-服务器之间的一般交互：请求/回复

无连接的协议：高效，受传输故障的影响，适合局域网

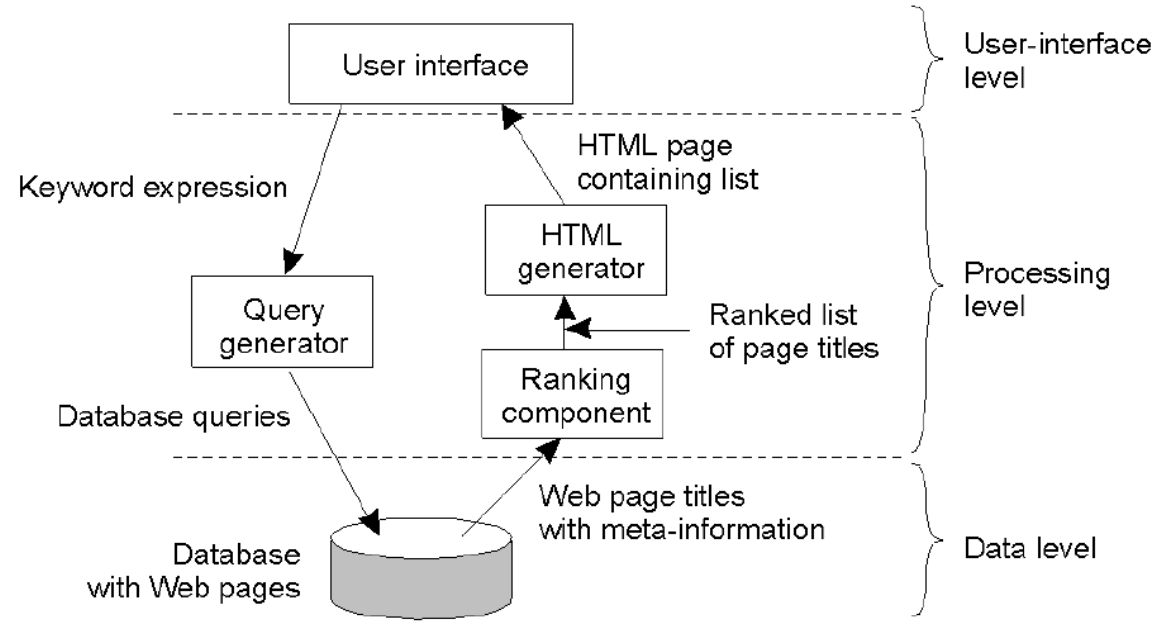
基于连接的协议：性能相对较低，适合广域网

客户服务器应用程序通常组织为三个层次：

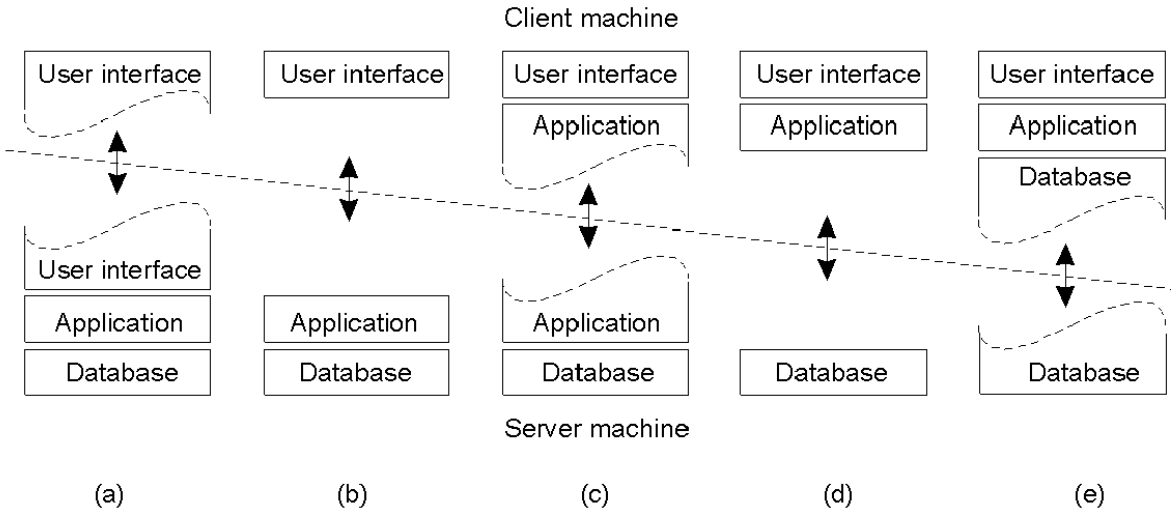
用户接口层：用户交互所需要的一切；

处理层：应用程序核心功能；

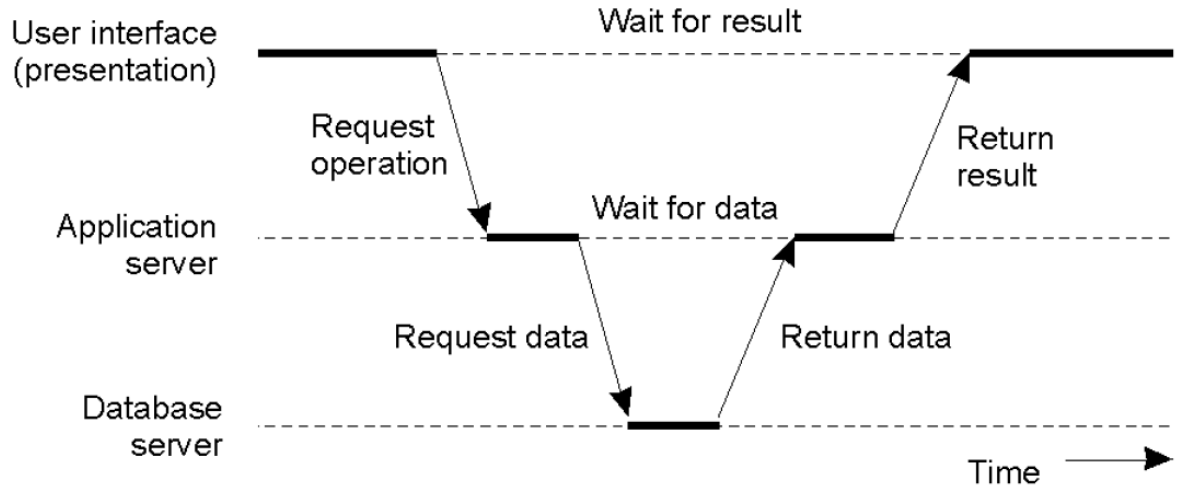
数据层：操作数据或文件系统，保持一致性。



物理两层体系结构



物理三层体系结构



第三章 分布式进程管理

1.进程和线程的比较

	进程	线程
地址空间和其他资源	进程间相互独立，同一进程的各个线程间共享该进程地址空间和其他资源，某进程内的线程在其他进程不可见	共享进程内部地址空间和其他资源。
通信	进程间通信通过IPC	线程间可以直接读写进程数据段（如全局变量）来进行通信--需要进程同步和互斥手段的辅助，以保证数据的唯一性。
调度	进程上下文切换慢	线程上线文切换速度快

结论：

- 多线程能提高性能
- 线程不像进程那样彼此隔离，并受到系统自动提供的保护，因此多线程应用程序开发需要付出更多努力。

2.多线程服务器的优点——简顺阻

- 显著简化服务器代码。
- 应用并发技术来开发高性能的服务器变得更加容易。
- 多线程能够保留顺序处理的思路，使用阻塞性系统的系统调用，仍然能到达并行处理的目的。
- 使用阻塞系统调用使编程更容易，并行处理能提高系统的性能。

3.代码迁移的动机有哪些?

- **实现负载均衡**：将进程从负载重的系统迁移到负载轻的系统，从而改善整体性能；
- **改善通信性能**：交互密集的进程可迁移到同一个节点执行以减少通信开销；当进程要处理的数据量较大时，最好将进程迁移到数据所在的节点。
- **可用性**：需长期运行的进程可能因为当前运行机器要关闭而需要迁移；
- **使用特殊功能**：可以充分利用特定节点上独有的硬件或软件功能。

4.进程对资源的绑定类型有哪些?

- 按标志符（URL）
- 按值
- 按类型

5.资源对机器的绑定类型有哪些?

- 未连接（数据文件）
- 附着连接（数据库）
- 紧固连接（本地设备）

资源对机器绑定

进程对资源绑定	未连接	附着连接	紧固连接
按标志符	MV (or GR)	GR (or MV)	GR
按值	CP (or MV, GR)	GR (or CP)	GR
按类型	RB (or MV, CP)	RB (or GR, CP)	RB (or GR)

- MV：移动资源
- GR：建立全局系统范围内引用
- CP：复制资源的值
- RB：将进程重新绑定到本地同类型资源

第四章 分布式系统通信

1.什么是远程过程调用？远程过程调用的步骤。

远程过程调用：当机器A上的进程调用B上的进程时，A上的调用进程被挂起，而B上的被调用进程开始执行。调用方可以通过使用参数将信息传送给被调用方，然后可以通过传回的结果得到信息。编程人员看不到任何消息传递过程。这种方法称为远程过程调用。

远程过程调用步骤：

1.客户过程以正常的方式调用客户存根
2.客户存根生成一个消息，然后调用本地操作系统
3.客户端操作系统将消息发送给远程操作系统
4.远程操作系统将消息交给服务器存根
5.服务器存根将参数提取出来，然后调用服务器
6.服务器执行要求的操作，操作完成后将结果返回给服务器存根
7.服务器存根将结果打包成一个消息，然后调用本地操作系统
8.服务器操作系统将含有结果的消息发送回客户端操作系统
9.客户端操作系统将消息交给客户存根
10.客户存根将结果从消息中提取出来，返回给调用它的客户过程

2.消息持久通信与暂时通信的区别?

持久通信：通信双方不必保持运行。

暂时通信：通信系统只在发送者和接收者运行时存储消息。

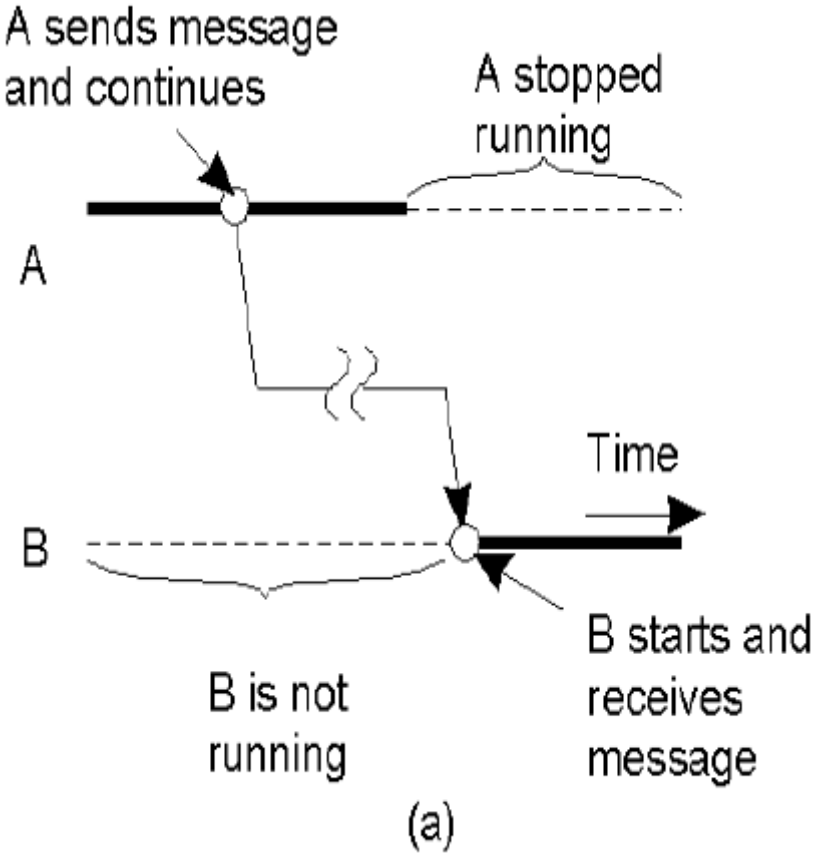
3.消息同步通信与异步通信的区别?

同步通信：等待消息的回复。

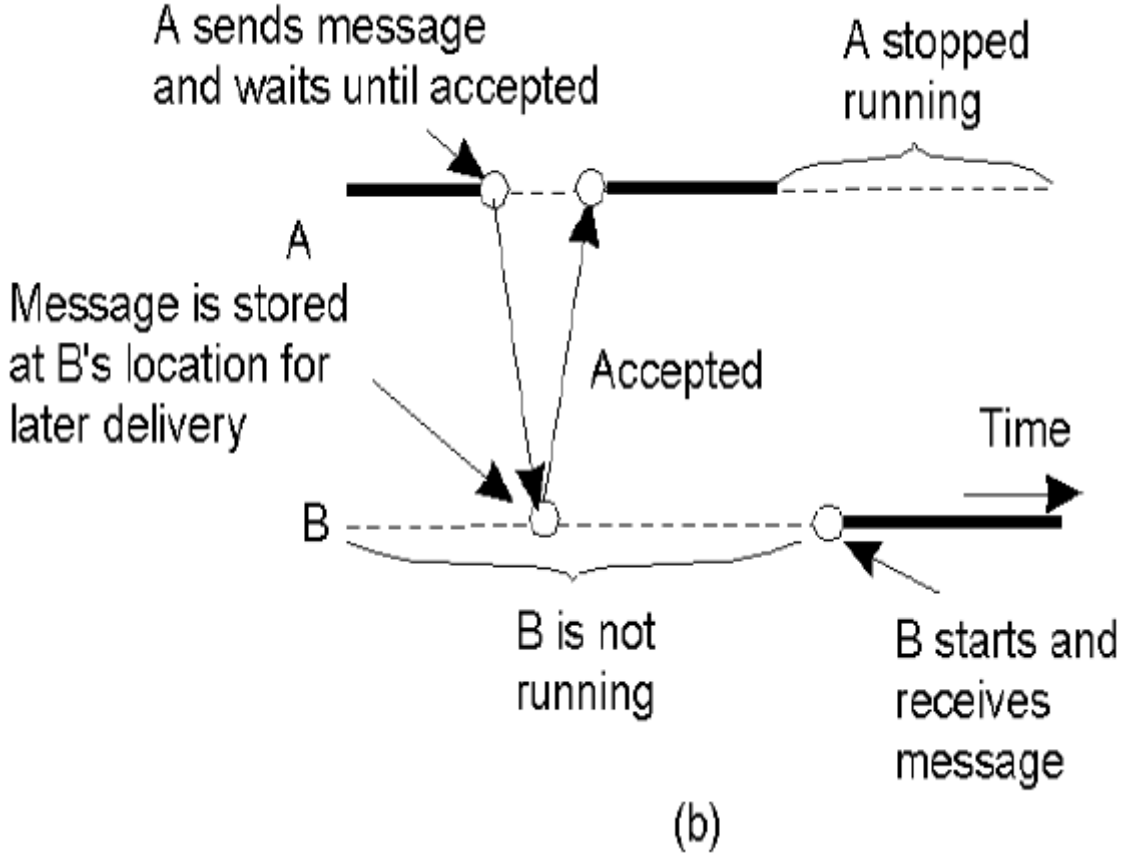
异步通信：不等待，立即执行其他程序。

4.能够判断消息通信的类型

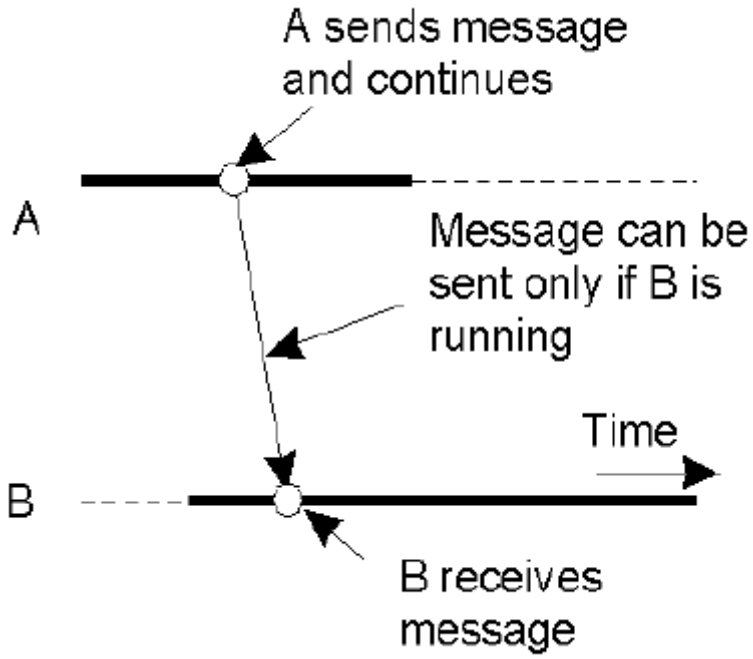
持久异步通信：提交消息后立即执行其他程序，电子邮件。



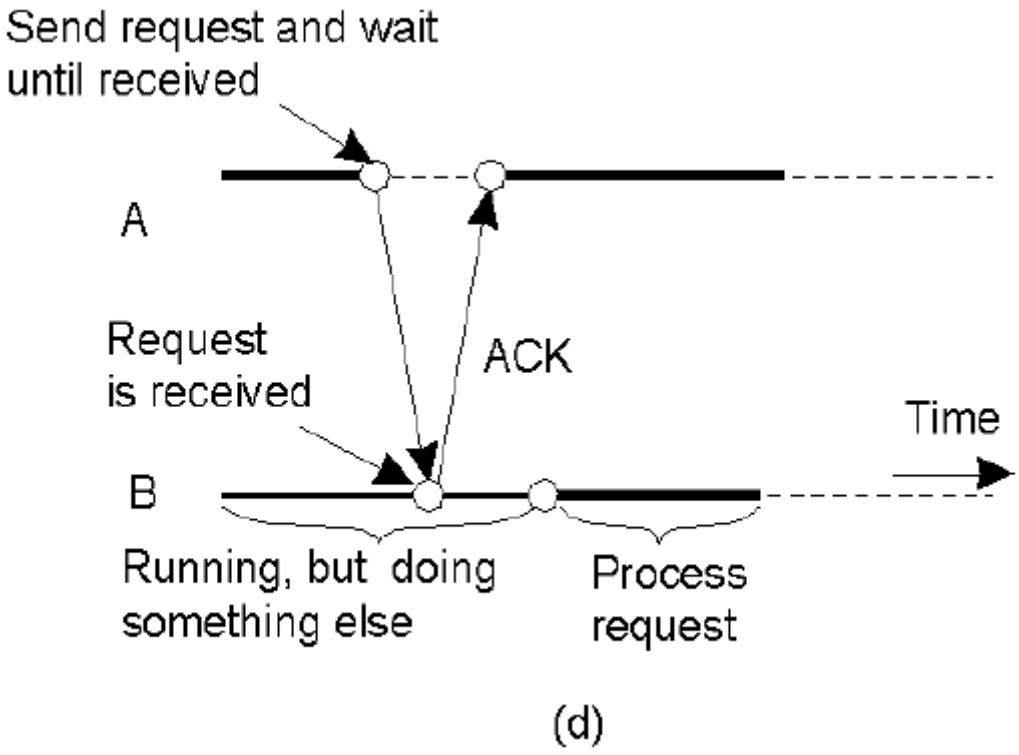
持久同步通信：提交消息后会被阻塞，直到消息已到达并存储在接收主机。



暂时异步通信：



基于接收的暂时同步通信：



(d)

5.多播通信：反熵和gossiping

- 反熵传播模型：

  服务器P周期的随机选取一台服务器Q交换更新，方式包括：

  - P只把自己的更新推入Q
  - P只从Q拉出新的更新
  - P和Q相互发送更新
  - 可以证明：如果初始只有一台服务器具有传染性，无论采取那种形式，更新最终将被传播到所有服务器上。
- Gossiping：

思想：

- 如果服务器P刚刚因为数据项x而被更新，那么它联系任何一个其他服务器Q，并试图将更新推入Q。
- 如果Q已经被其他服务器更新了，P可能会失去继续扩散的兴趣，变成隔离的（这种可能性是1/k）

这种一种快速传播更新的方法，但不能保证所有的服务器都被更新了，可以将gossiping和反熵模型结合。

第五章 命名

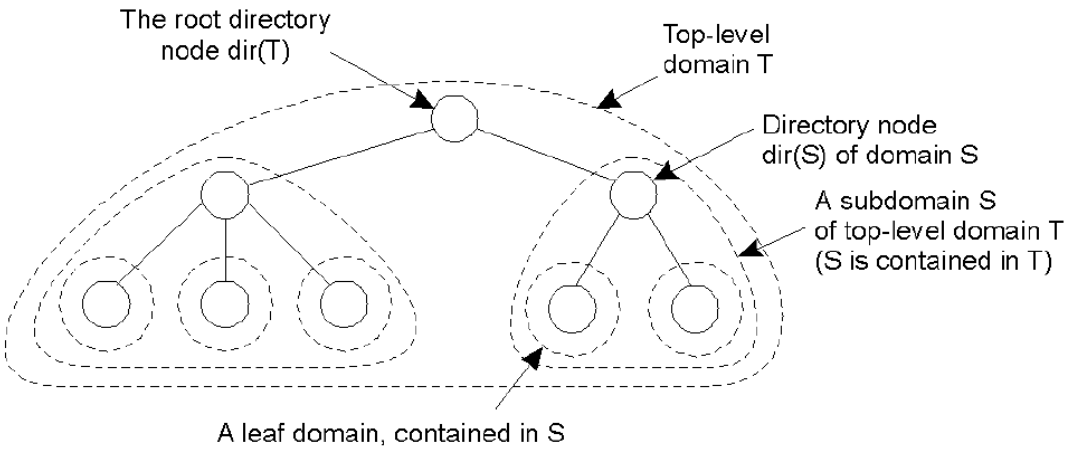
1.移动实体定位的方法有哪些?

- 简单方法
  - 多播
  - 广播
  - 转发指针

    当实体A移动到B时，它将在后面留下一个指针，这个指针指向它在B中的新位置。这种方法的主要优点是它很简便：一旦找到实体以后，客户就可以顺着转发指针形成的链来查找实体的当前地址。
- 基于起始位置的方法

  每个移动主机都使用一个固定IP地址，所有与该IP地址进行的通信一开始都被转发到移动主机的宿主代理中。宿主代理位于局域网中，与包含在移动主机IP地址中的网络地址相对应。当一台移动主机转移到另一个网络中时，它会请求一个可以用来通信的临时地址。这种转交地址要在宿主代理中注册。
- 分层方法

  网络被划分为一组域，目录节点用来记录域包含的实体，叶域的目录节点N记录实体E在域中的位置，更高一层域的目录节点N'记录实体E的位置，包含指向N的指针。



2.描述分层方法中查找一实体的过程。

希望定位实体E的客户向他所在的叶域D的目录节点发送一个查找请求，如果这个目录节点没有存储该实体的位置记录，那么就说明该实体现在不在D中。因此，这个节点会把请求转发给他的父结点。注意，父结点代表一个比它的子域更大的域。如果父结点也没有E的位置记录，那么就会把该查找请求转发给更高一层的域，以此类推。

如果结点M存储了E的位置记录，那么一旦请求到达M后，就可以知道E位于结点M的域dom（M）中。M中存储了一条位置记录，其中包含一个指向其子域的指针。然后，把请求转发给那个子域的目录节点，那个子域会依次进一步向树的下方转发请求，直到请求最终到达叶节点为止。存储在叶结点中的位置记录会包含E在哪一个叶域中的地址。这样就可以把这个地址返回发起请求的客户。

3.描述分层方法中插入一实体的过程。

假设实体E在叶域D中创建了一个副本，就需要插入这个副本的地址。插入操作从D的叶结点dir(D)开始，然后D会立即把插入请求转发给它的父结点。父结点同样会转发插入请求，直到插入请求到达已经为E存储了位置记录的目录结点M为止。

然后，结点M会在E的位置记录中存储一个指针，该指针指向转发插入请求的那个子结点。此时，该子结点会建立一条关于E的位置记录，该位置记录中包含一个指针，指向转发请求的下一层结点。这个过程会持续进行，直至到达发起请求的叶结点为止。最后，那个叶结点会创建一条记录，这条记录包含实体在相关叶域中的位置。

第六章 同步

1.Lamport时间戳算法的思想

- 网络上的每个系统（站点）维护一个计数器，起时钟的作用
- 每个站点有一个数字型标识，消息格式为（m, Ti, i），m为消息内容，Ti为时间戳，i为站点标识
- 当系统发送消息时，将时钟加一
- 当系统j接收消息时，将它的时钟设为当前值和到达的时间戳这两者的最大者加一
- 在每个站点，时间的排序遵循一下规则
    - 对来自站点i的消息x和站点j的消息y，如果
        - Ti<Tj或
        - Ti=Tj，且i<j
    - 则说消息x早于消息y

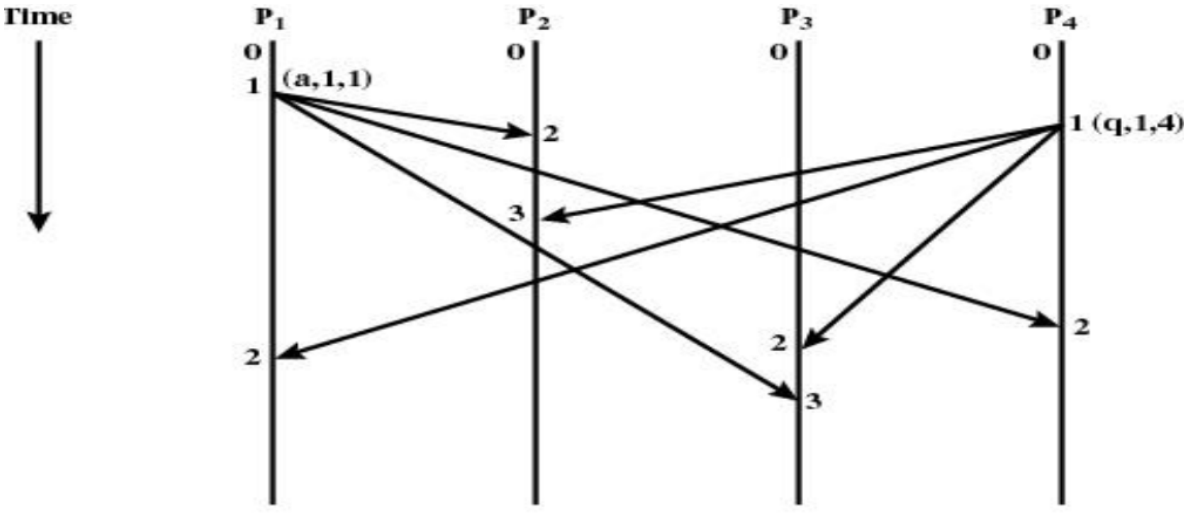


Figure 14.9 Another Example of Operation of Timestamping Algorithm

哪个事件在实际上首先发生并不重要，重要的是所有进程对事件的发生顺序意见一致。

2.选举算法中Bully算法的思想

当进程P注意到需要选举一个进程做协调者时：

- 向所有进程号比它高的进程发ELECTION消息
- 如果得不到任何进程的响应，进程P获胜，成为协调者
- 如果有进程号比它高的进程响应，该进程接管选举过程，进程P任务完成
- 当其他进程都放弃，只剩一个进程时，该进程成为协调者
- 一个以前被终止的进程恢复后也有选举权

3.选举算法中环算法的思想

- 不使用令牌
- 按进程号排序，每个进程都知道自己的后继者
- 当进程P注意到需要选举一个进程作协调者时：
    - 就创建一条包含该进程号的ELECTION消息，发给后继进程
    - 后继进程再将自己的进程号加入ELECTION消息，依次类推
    - 最后回到进程P，它再发送一条COORDINATOR消息到环上，包含新选出的协调者进程（进程号最大者）和所有在线进程

4.实现事务的方法

两种实现事务的方法：

1. 私有工作空间
    - 为进程提供一个私有工作空间，包含进程要访问的所有对象
    - 进程的读写操作在私有工作空间进行，而不对实际的文件系统进行
    - 开销大，可以进行优化使之可行
        - 读操作不复制
        - 写操作时复制
    - 如果事务终止，私有工作空间被释放，指向的私有块被删除
    - 如果事务提交，私有索引被移到父辈空间，不再被访问的块被释放掉
2. 写前日志
    - 先写日志，再做实际修改
    - 日志内容：哪个事务在对文件进行修改，哪个文件和数据被改动，新值和旧值是什么
    - 日志写入后，改动才被写入文件
    - 事务终止，使用写前日志回退到原来的状态
    - 借助稳定存储器中的写前日志：当系统崩溃后，完成事务或取消事务

5.分布式的死锁检测Chandy-Misra-Haas算法的思想

允许进程一次请求多个资源，而不是一次一个。

- 通过允许多个请求同时进行是的事物的增长阶段加速
- 这使得一个进程可以同时等待两个或多个进程

当某个进程等待资源时，例如P0等待P1将调用Chandy-Misra-Haas算法。

- 生成一个探测消息并发送给占用资源的进程
    - 消息由三个数字构成：阻塞的进程，发送消息的进程，接受消息的进程。
    - 由P0到P1的初始消息包含三元组（0，0，1）.
- 消息到达后，接受者检查以确认他自己是否也在等待其他进程。
    - 若是，就更新消息，字段1保持不变，字段2改成当前进程号，字段3改为等待的进程号。

    - 然后消息接着被发送到等待的进程。
    - 若存在多个等待进程，就要发送多个不同的消息。
- 不论资源在本地还是在远程，该算法一直继续下去。
- 若消息转了一圈后又回到最初的发送者，即字段1所列的进程，就说明存在一个有死锁的环路存在。

第七章 一致性和复制

1.复制的目的和代价

目的：

- 可靠性
- 性能
    - 服务器数量扩展
    - 地理区域扩展

代价：一致性

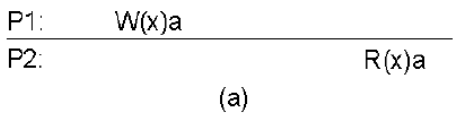
- 网络通信开销
- 强一致性要求的原子操作很难快速完成

2.能区分是否符合严格一致性、顺序一致性、因果一致性和FIFO一致性

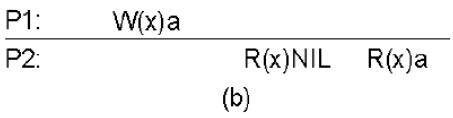
- **严格一致性**

  任何对数据项X的读操作将返回最近一次对X进行写操作的值

  对所有进程来说，所有写操作都是瞬间可见的，系统维护着一个绝对的全局时间顺序



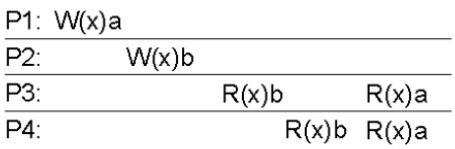
(a)



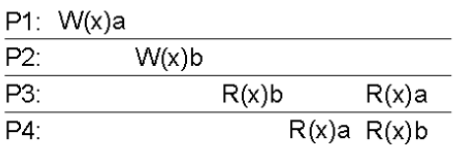
(b)

- **顺序一致性**

  顺序一致性对存储器的限制比严格一致性要弱一些，要满足以下的条件：
  （1） 每个进程的内部操作顺序是确定不变的；
  （2） 假如所有的进程都对某一个存储单元执行操作，那么，它们的操作顺序是确定的，即任一进程都可以感知到这些数据同样的操作顺序。



(a)

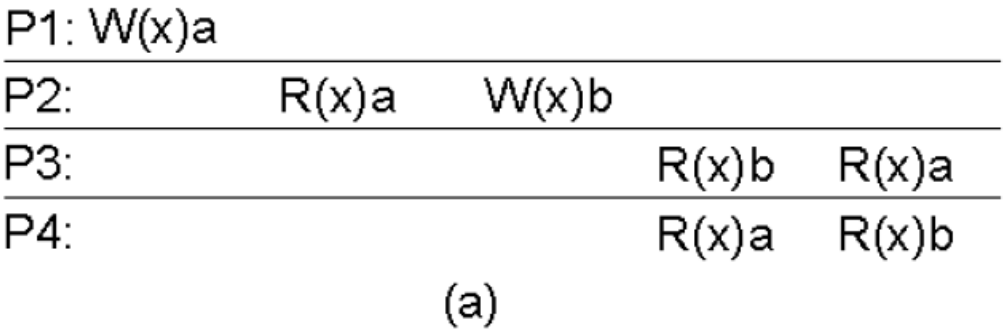


(b)

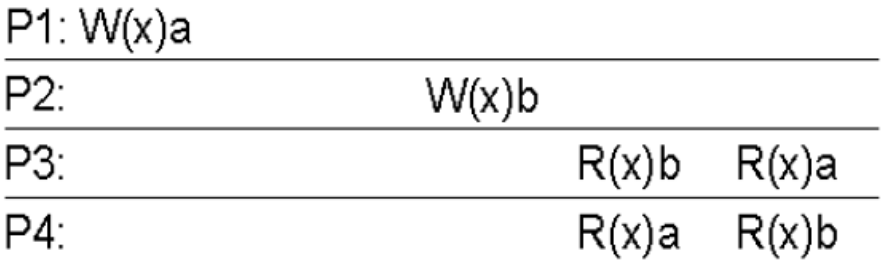
- **因果一致性**

  所有进程必须以相同的顺序看到具有潜在因果关系的写操作

  不同机器上的进程可以以不同的顺序看到并发的写操作



(a)

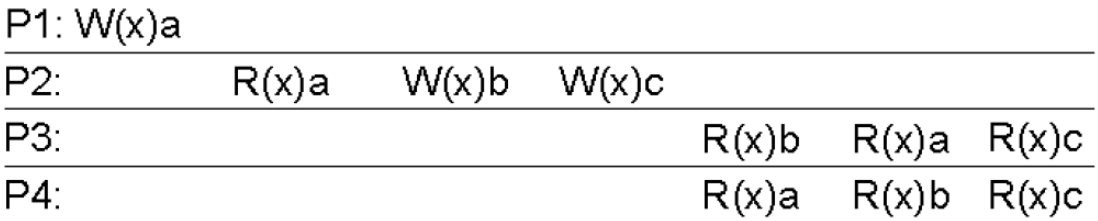


(b)

a不符合因果一致性，P2进程的操作只能在P1之后发生，因此如果有进程读到了R(x)b那么它不是不可能再读到R(x)a的。

- **FIFO一致性**

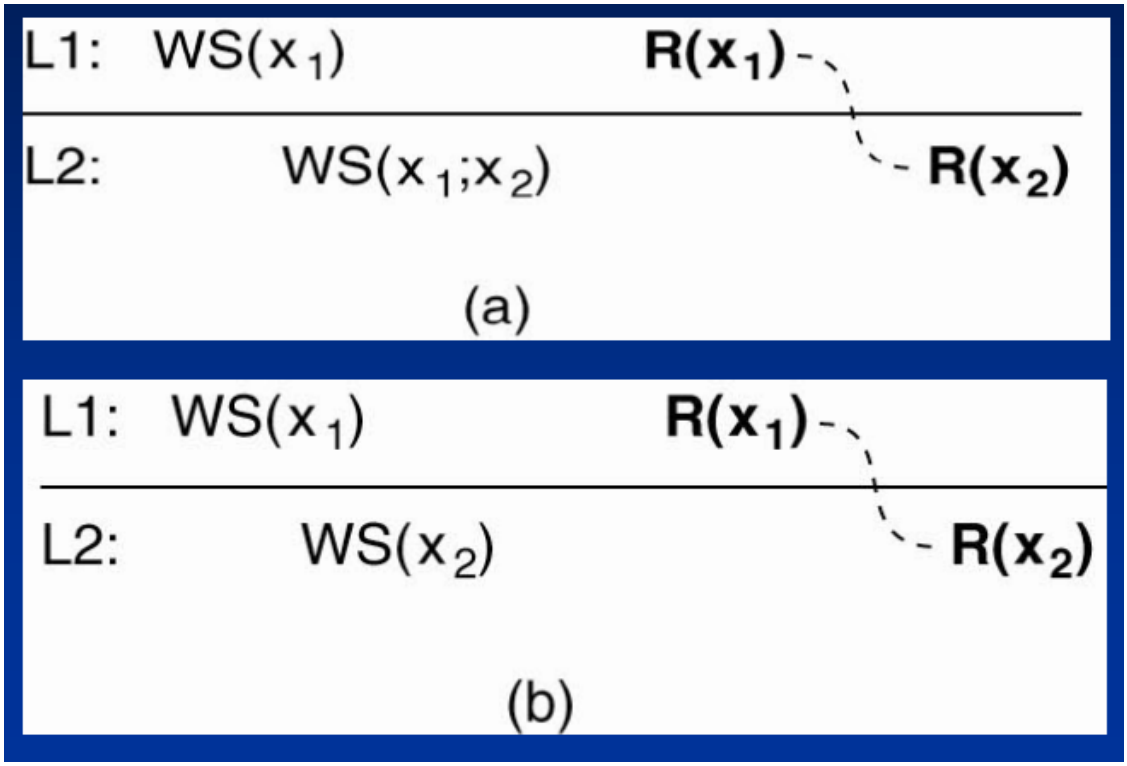
  由某一个进程完成的写操作可以被其他所有的进程按照顺序感知到，而从不同进程中来的写操作对不同的进程可以有不同的顺序。



只要保证P2进程中的写数据的顺序即可，一定是先读到b再读到c才正确。

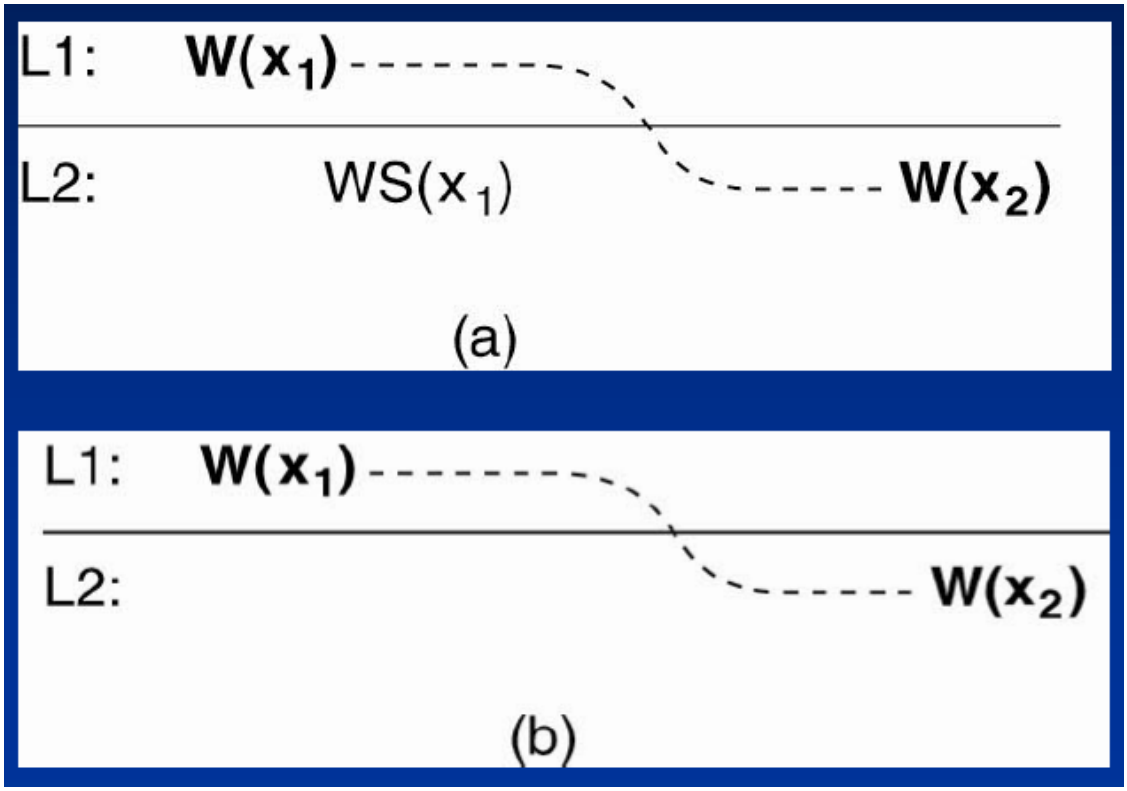
3.能区分是否符合单调读、单调写、写后读和读后写

- **单调读**



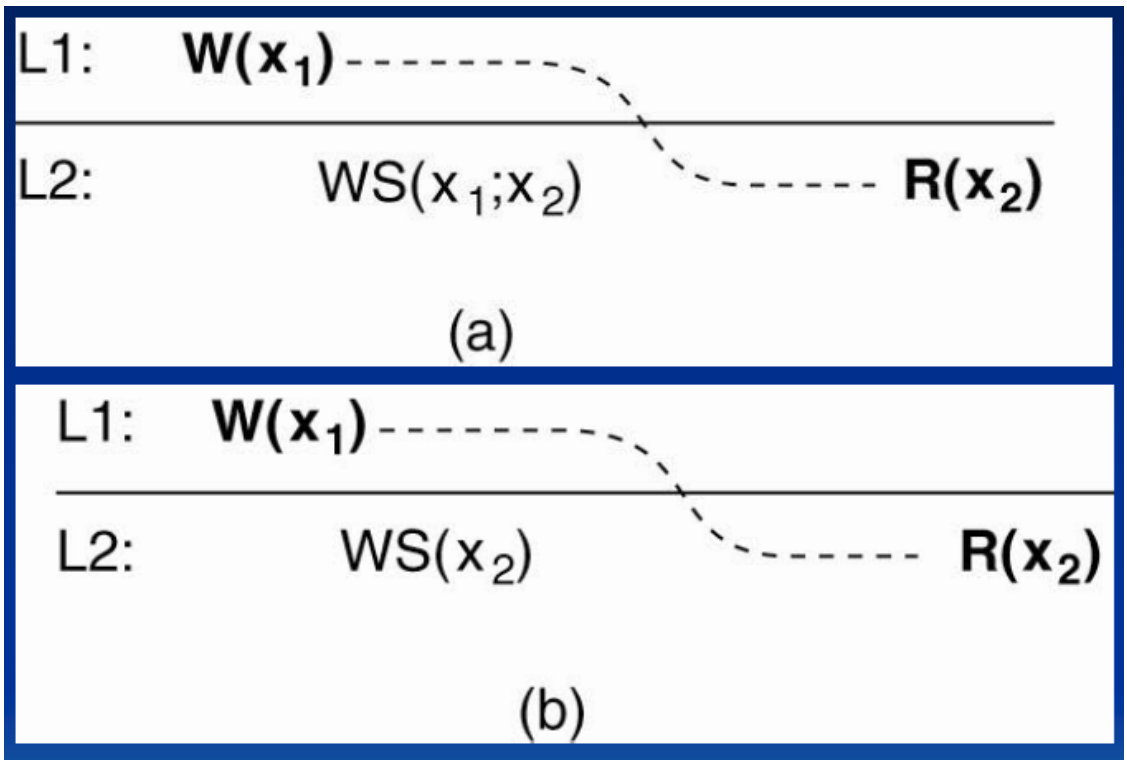
a满足单调读

- **单调写**



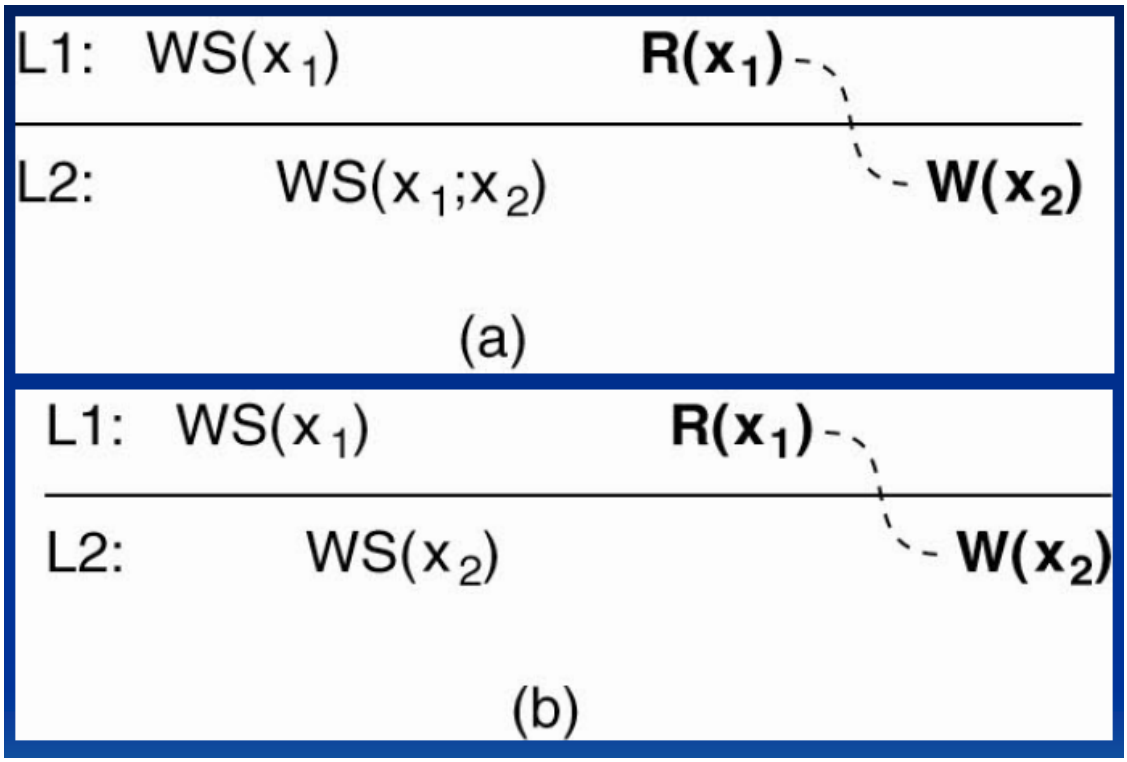
a满足单调写

- **写后读**



a满足写后读

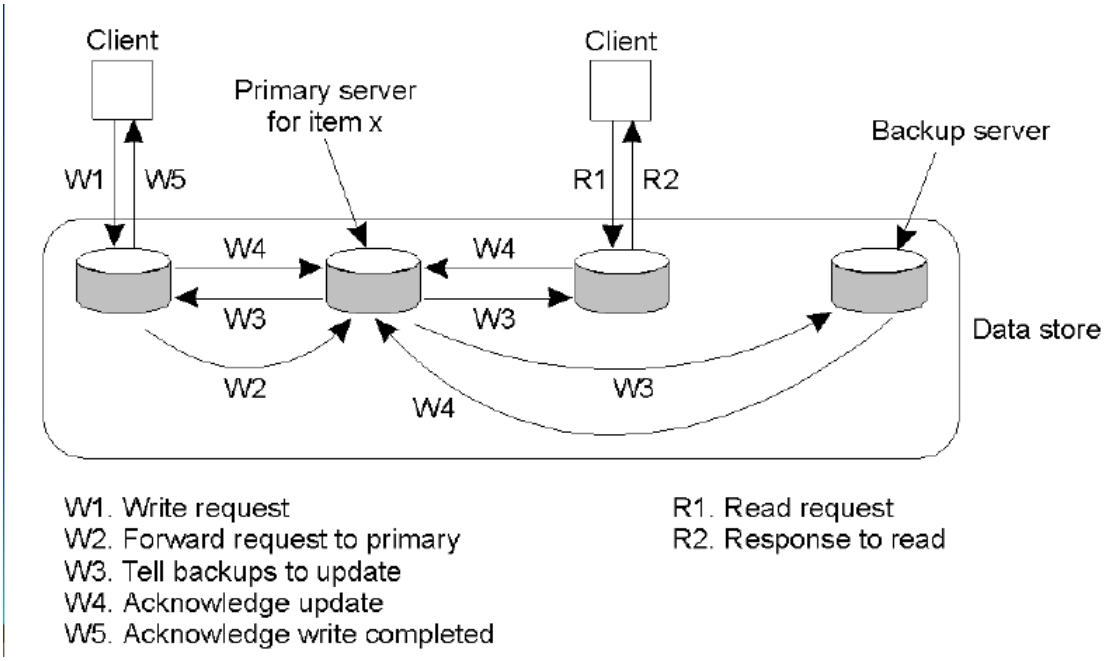
- **读后写**



4.基于主备份的协议：远程写协议、本地写协议

- **远程写协议**

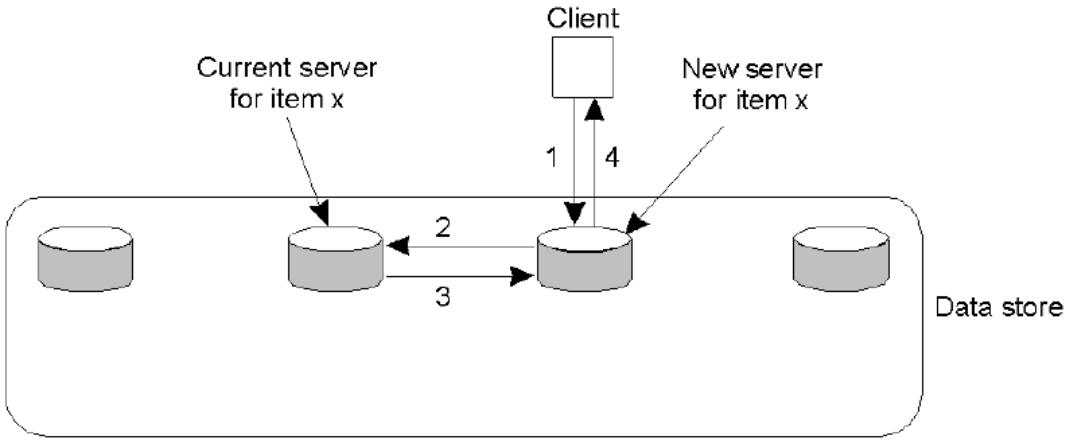
  基于主备份的远程写协议，所有读操作和写操作都被转发到一个固定的服务器上



- **本地写协议**

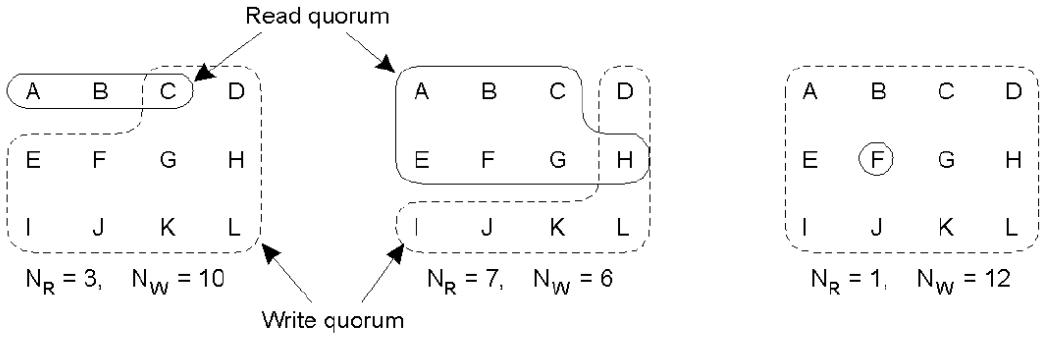
  基于主备份的本地写协议，其中一个单一的拷贝在多个进程间移动，保证一致性，需要跟踪数据项的当前位置，广播、转发指针、基于原始位置的方法和层次定位服务。

  主备份移动到要执行更新的进程那里。



5.复制的写协议：主动复制、基于法定数目的协议

- 主动复制
  - 写操作可以在多个副本上执行，每个副本对应一个进程，该进程执行更新操作
  - 写操作导致更新传播
  - 操作需要在各地按相同的顺序进行
    - 时间戳
    - 定序器
- 基于法定数目的协议
  - 客户在读写一个复制的数据时，先向多个服务器提出请求，获得许可；
  - 读团体N_R和写团体N_W，N个副本。
  - $N_R + N_W > N; N_W > \frac{N}{2}$



a,c正确，b有可能产生冲突。

第八章 容错性

1.什么叫容错性

容错意味着系统即使发生故障也能提供服务。

容错与可靠性相联系，包含以下需求：
–可用性（Availability）：任何给定的时刻都能及时工作
–可靠性（Reliability）：系统可以无故障地持续运行
–安全性（Safety）：系统偶然出现故障能正常操作而不会造成任何灾难
–可维护性（Maintainability）：发生故障的系统被恢复的难易程度

2.拜占庭将军问题

拜占庭将军问题是用来描述分布式系统一致性问题。

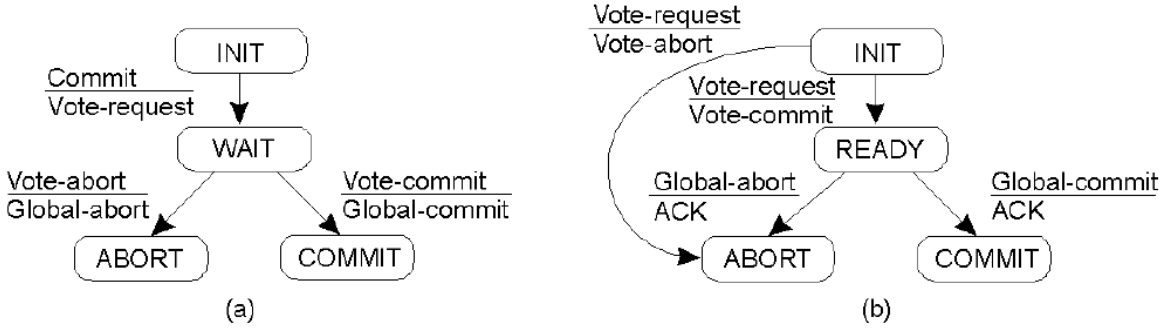
系统中存在故障进程，发出错误信息或不发出信息，导致系统一致性出现问题，称为拜占庭将军问题。

3.什么叫原子多播

- 需要在存在进程失败的情况下获得可靠多播的情况
- 原子多播
  - 消息要么发送给所有进程，要么一个也不发送
  - 通常需要所有的消息都按相同的顺序发送给所有的进程
  - 分布式系统中的复制数据库
  - 原子多播确保没有故障的进程对数据库保持一致；当一个副本从故障中恢复并重新加入组时，原子多播强制它与其他组成员保持一致。

4.分布式提交——两阶段提交的思想

简单、实用、可靠，成为事实上的工业标准。 在两段提交协议中，将提交分成两个阶段，
·第一阶段（表决阶段），事务的协调者询问各个参与者是否可以提交，此时，各个参与者将回答消息发给协调者。协调者根据收到的消息，看是否可以真正提交。
·第二阶段（完成阶段），如果可以提交，则通知各参与者立即执行提交，否则，通知它们中止此事务。



a)2PC中的协调者的有限状态机
b)2PC中的参与者的有限状态机
参与者一旦投票，则失去自主能力，必须等待协调者的最终决定，可能造成阻塞
可能的阻塞状态：
•参与者在INIT状态等待协调者的VOTE_REQUEST消息
•协调者在WAIT状态等待来自每个参与者的表决
•参与者在READY状态等待协调者发送的全局表决消息

第九章 分布式安全

1.什么是机密性和完整性?

机密性：系统将信息只向授权用户公开；

完整性：对系统资源的更改只能以授权方式进行；

2.对称加密系统和公钥系统的区别?

对称加密系统：加密与解密密钥相同，即$P = D_k(E_k(P))$，也称为共享密钥系统。

非对称加密系统：加密与解密密钥不同（一个公开、一个保密），但构成唯一的一对，即$P = D_{KD}(E_{KE}(P))$，也称为公钥系统。公钥加密系统只能加密短信息，太长的信息，加密时间太长。

3.什么是安全通道?

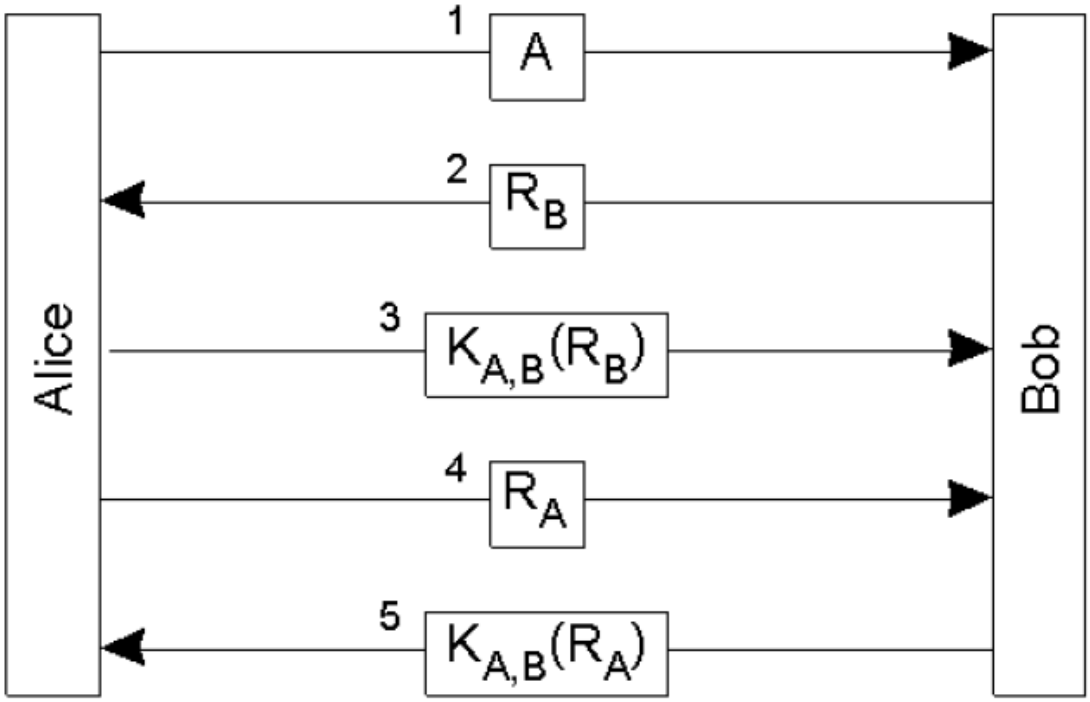
安全通道：使客户与服务器之间的通信保持安全，免受对消息的窃听、修改和伪造的攻击。

身份验证：通信双方需要验证身份。

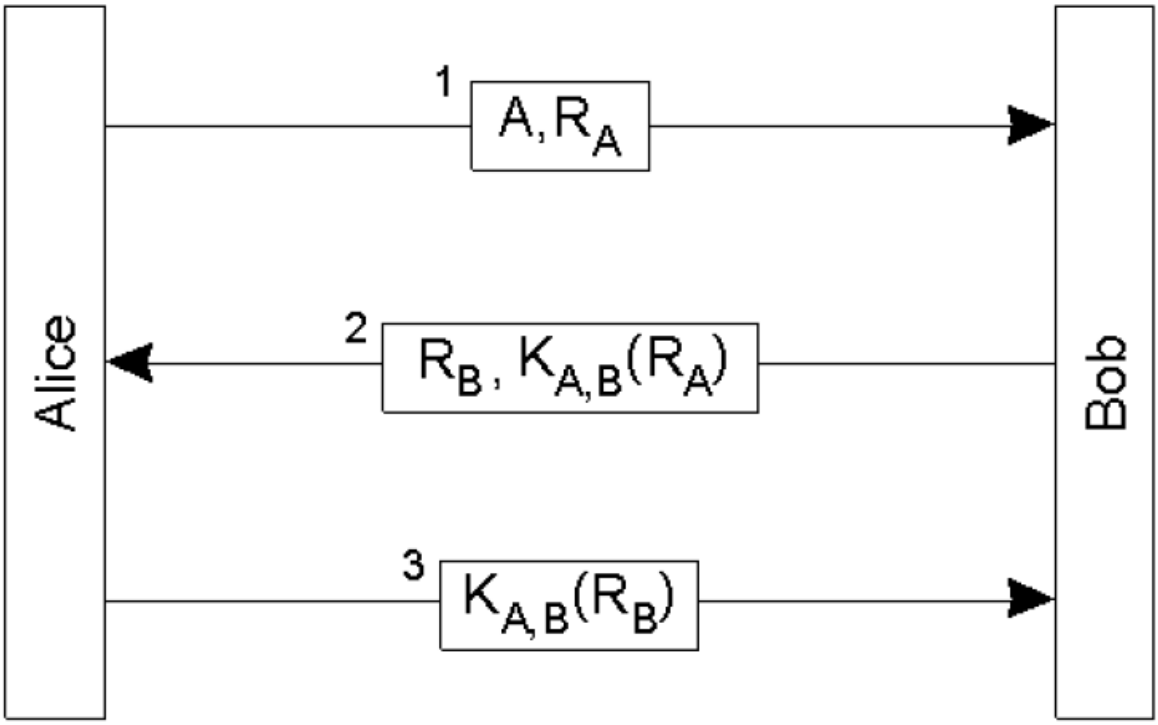
消息的完整性和机密性：消息未受到窃听、修改和伪造的攻击。

4.阐述基于共享密钥的身份验证的思想

质询-响应协议：一方向另一方质询一个响应，只有对方直到共享密钥时才能给与正确的响应。



基于共享密钥的身份验证，用三个消息代替五个



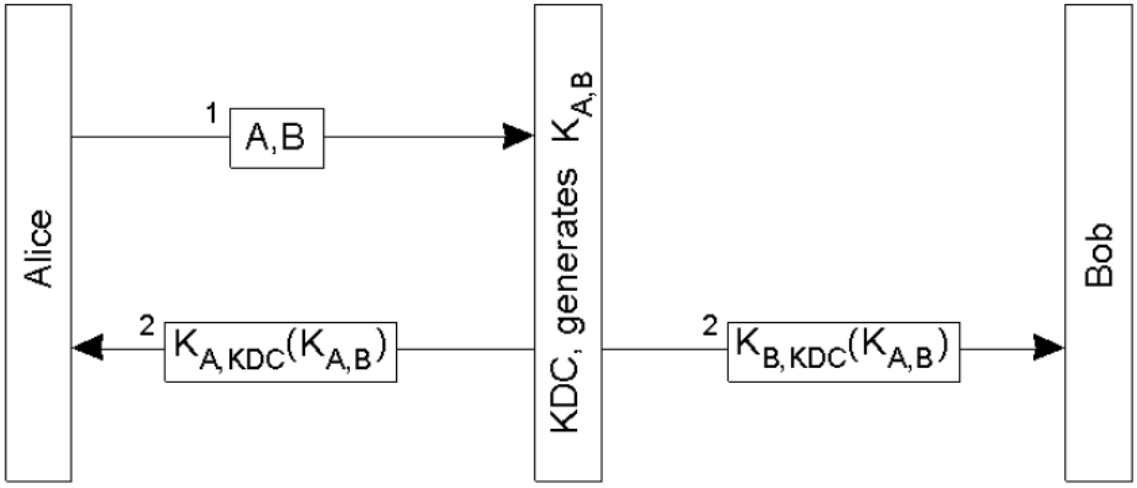
这样的方式很容易受到反射攻击。

5.阐述使用密钥发布中心的身份验证的思想

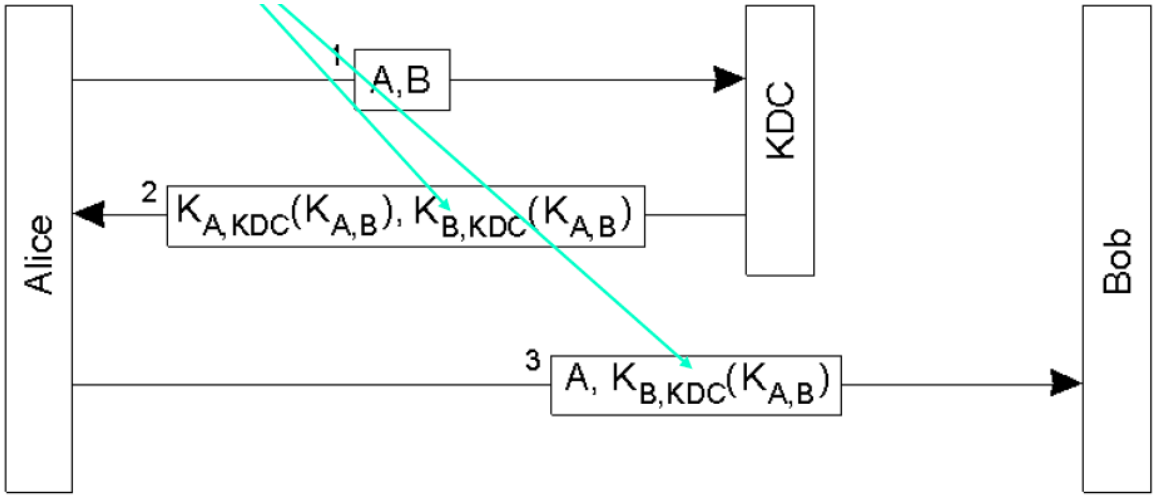
基于共享密钥的身份验证存在可拓展性问题：N台主机，需要N*(N-1)/2个密钥。

使用KDC（key distribution center）只需要管理N个密钥。

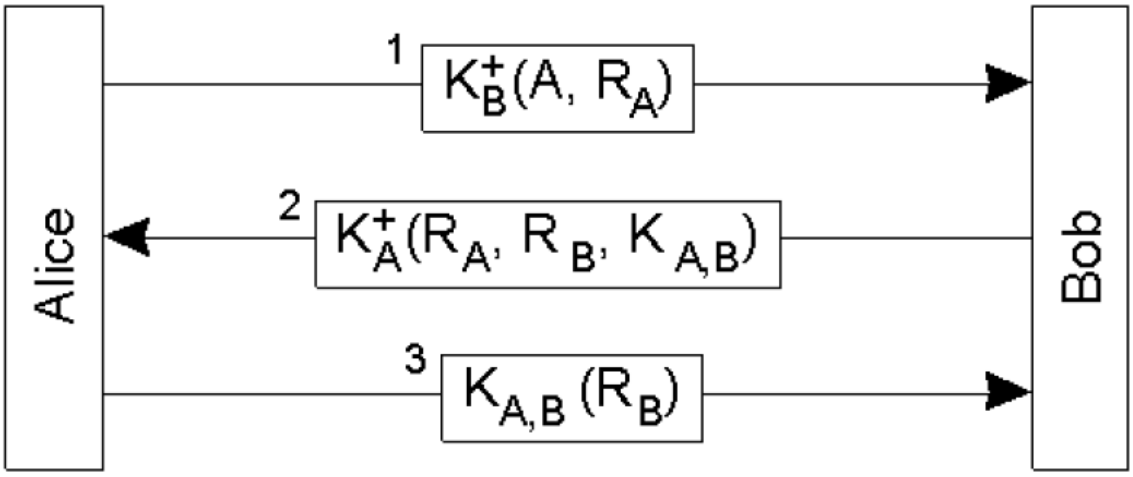
KDC与每台主机共享一个密钥；向通信的两主机分发一个密钥通信。



使用票据的改进型密钥发布中心的身份验证



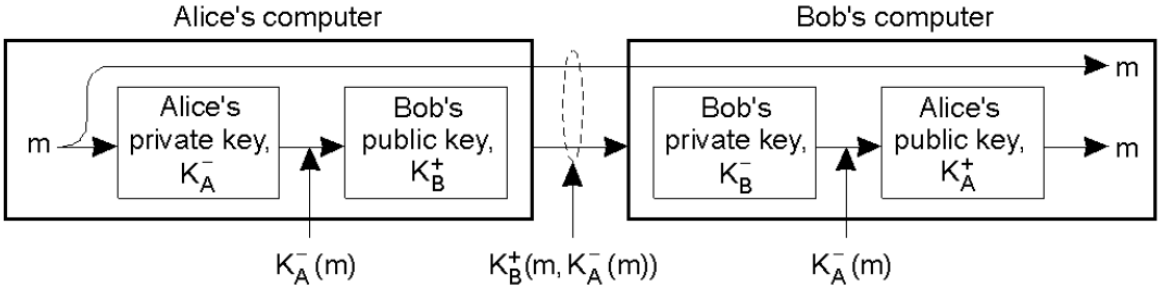
6.阐述使用公钥加密的身份验证的思想



当Alice想同Bob进行交谈时，会使用Bob的公钥对信息进行加密，加密的信息包含一个质询R_A，当Bob接收到Alice发送的消息之后，只有Bob可以通过自己的私钥对该进行进行解密。之后为了保证Alice知道对方是Bob，则Bob使用Alice的公钥加密信息，该信息包含Alice发送的质询R_A与Bob本身的质询R_B，与后续建立会话的加密共享密钥$K_{A,B}$，Alice收到消息之后就能确定另一方是Bob，随后通过共享密钥发送Bob的质询信息R_B，Bob接收到质询信息之后进行解密，就能确定对方是Alice，至此Alice与Bob之间的安全通信就建立起来了。

7.使用公钥加密对消息进行数字签名的思想

身份验证保证了信息的机密性，但是信息的完整性（信息不可随意修改）没有得到保证，因此一种主流的方式是使用公钥加密对消息进行数字签名。



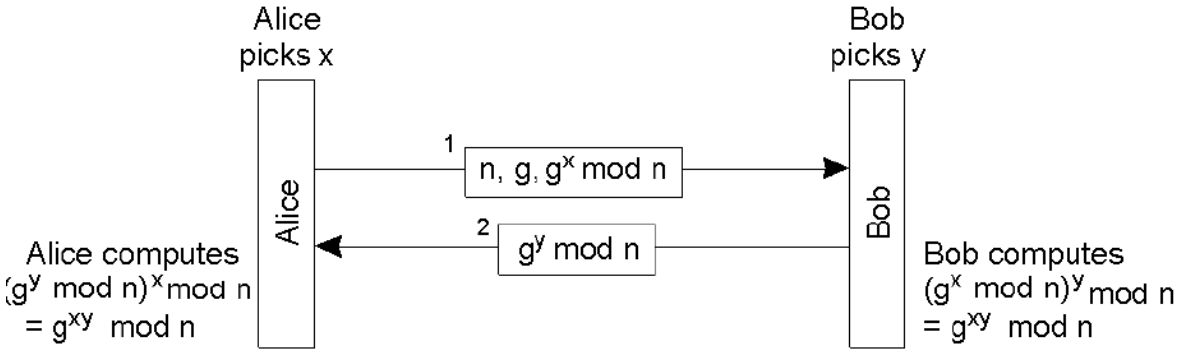
Alice向Bob发送消息m时，先使用自己的私钥对消息进行加密，生成该消息的数字签名，随后还可以使用Bob的公钥来对该消息进行加密，随后传输到Bob处，Bob使用自己的私钥和Alice的公钥对消息逐层解密，得到Alice数字签名的信息，随后与传输来的信息m进行比较，如果一致，则证明经过Alice数字签名的消息是完整的，没有经过修改的。

- Bob通过比较m与其解密版确定m是否来自Alice
- Bob保留m的签名版：
  - 防止Alice的否认。
  - 防止Bob对消息的恶意修改：它必须证明修改版本也是由Alice签名的。

8.Diffie-Hellman建立共享密钥的原理

首先，Alice和Bob双方约定2个大整数n和g,其中1<g<n，这两个整数无需保密，然后，执行下面的过程：

- Alice随机选择一个大整数x (保密)，并计算$X = g^x \mod n$
- Bob随机选择一个大整数y (保密)，并计算$Y = g^y \mod n$
- Alice把X发送给B,B把Y发送给ALICE
- Alice计算$K = Y^x \bmod n = (g^y \bmod n)^x \bmod n = g^{xy} \bmod n$
- Bob计算$K = X^y \bmod n = (g^x \bmod n)^y \bmod n = g^{xy} \bmod n$
- K即是共享的密钥。
- 监听者在网络上只能监听到X和Y，但无法通过X，Y计算出x和y，因此，无法计算出K。



9.权能和委派

权能：

权能是对于指定资源的一种不可伪造的数据结构，它确切指定它的拥有者关于该资源的访问权限。

为了调用一个对象的操作，客户必须将权能传给服务器检查。

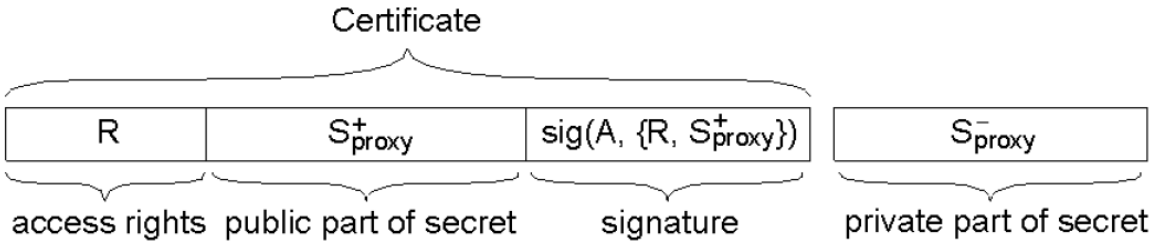
服务器创建对象时，客户得到的是所有者权能（全1），check字段是随机选择的，同时存储在权能和服务器中的一个表中。

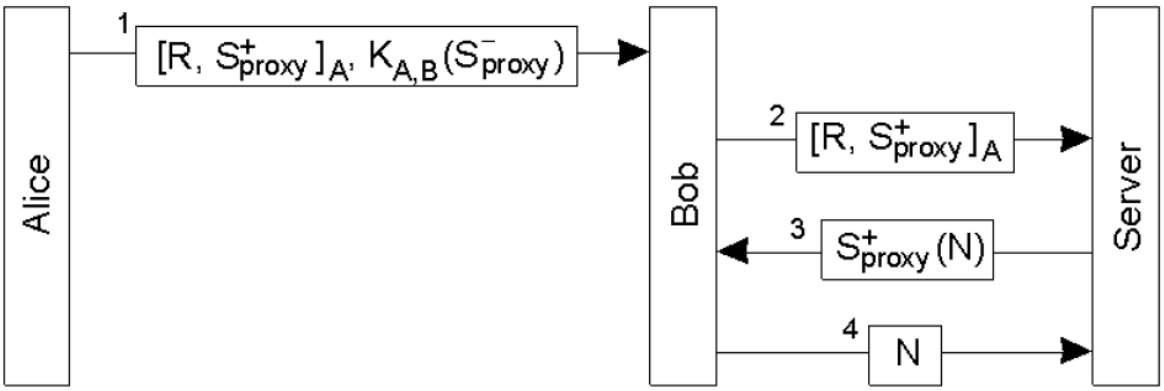
客户可以从一个所有者权能生成一个受限权能，并发送给另一个进程。

委派：

将某些访问权限从一个进程传递给另一个进程，Alice可以构造证书：

- Bob具有权限R
- 此证书的持有者具有权限R





第十章 分布式文件系统

1.NFS的共享预约

实现思想：

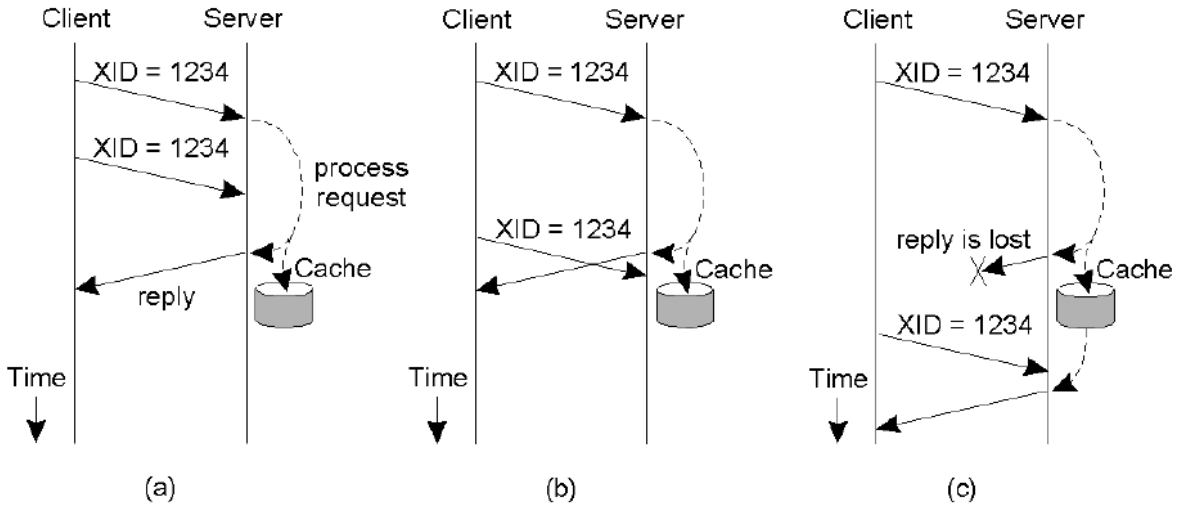
- 基本思想：让客户机和服务器的任何一个集合共享一个公用FS，NFS允许一台计算机既是客户机又是服务器。
- 当一个服务器输出某个目录时，该目录为根的子目录树同时被输出。服务器输出的目录列标记在文件的/etc/export中。
- 客户通过安装的方式访问服务器的输出目录。
- 基本特征：服务器输出目录，客户从远处安装他们。
- 优点：多个客户机同时安装同一个目录时，他们可以通过共享公用目录者的文件来进行通信。

2.NFS服务器的重复请求高速缓存

NFS的底层RPC不能保证可靠性，而且缺乏对重复请求的检测。

NFS服务器提供重复请求高速缓存解决：XID事务处理标识符。

重复性检查是为那些不能在两次执行中返回同一结果的操作而提供的。这方面经典的例子是rm命令。第一个rm命令也许成功了，但是如果应答丢失了，客户机将会重发这个命令。我们希望这样的重复请求能获得成功，此时重复高速缓存被查询，如果发现是一个重复请求则相同的（成功的）结果被作为第二个重复请求的结果返回，就好像是由第一个请求所产生的结果一样。

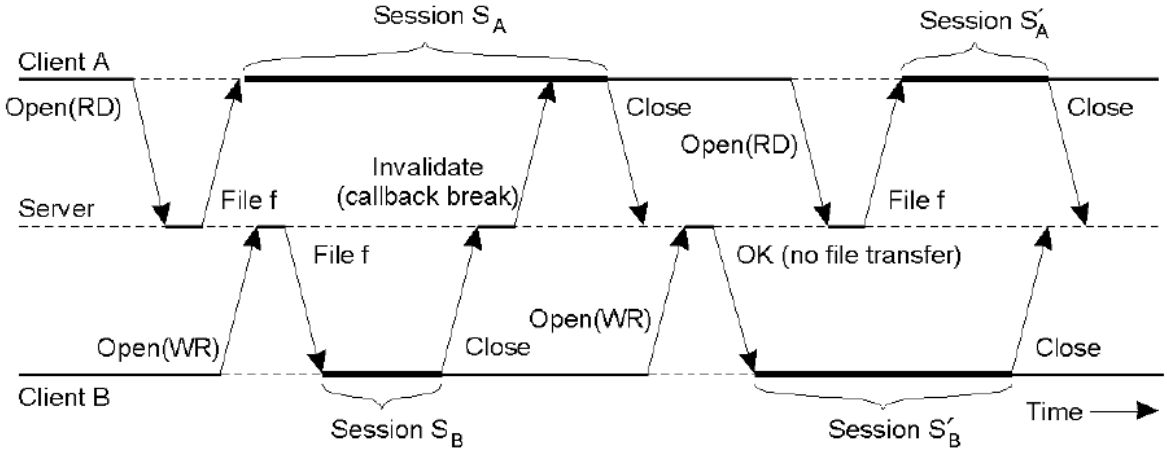


3.Coda的回叫承诺

回叫承诺：服务器记录哪些客户在本地缓存了文件的拷贝

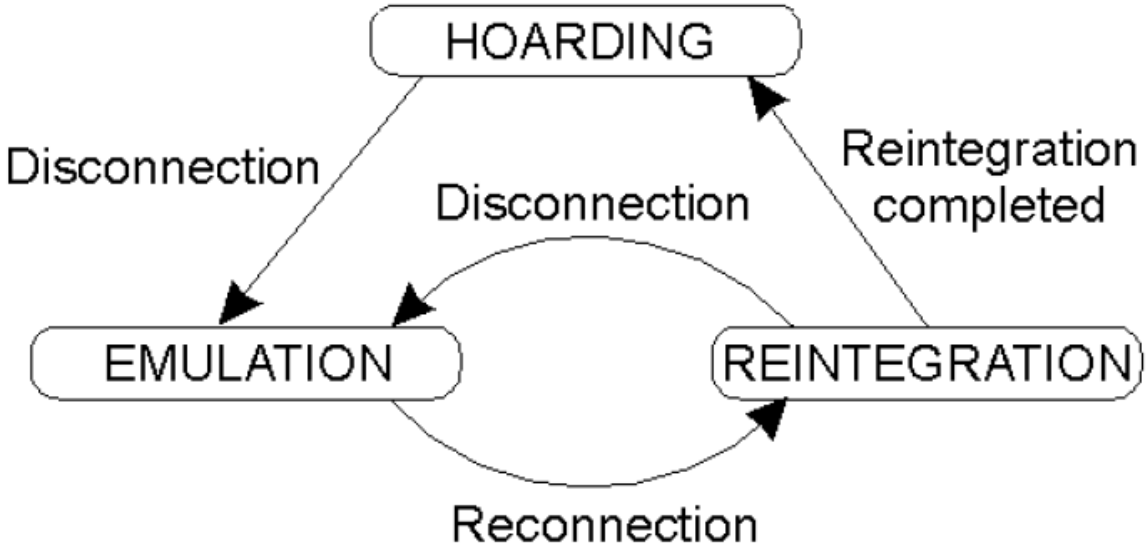
如果文件被客户更改，会通知服务器，后者向其他客户发无效化消息（回叫中断：服务器废弃回叫承诺）

如果客户在服务器上有未被废弃的回叫承诺，它就可以安全地在本地访问文件。



4.Coda的储藏技术

Coda允许客户在断开连接时（AVSG为空）的继续操作。 使用本地备份，再次连接后回传服务器。 基于事实：两个进程打开相同的文件进行写操作很罕见。 使用储藏技术（hoarding）。


HOARDING
Disconnection
Disconnection
Reintegration
completed
EMULATION
REINTEGRATION
Reconnection