**分布式考试题整理(2004-2010)**

*张航@WS Group*

**一、名词解释**

GPL General Public License 通用公共许可

ACM Association for Computing Machinery 美国计算机协会

IEEE institute of electrical and electronics engineers 电气与电子工程师协会

NFS Network File System 网络文件系统

DOS Distributed Operating System 分布式操作系统

CGI Common Gateway Interface 通用网关接口

ATM Asynchronous Transfer Mode 异步传输模式

POSIX Portable Operating System Interface for UNIX UNIX下的可移植操作系统接口

CSP commercial service provider 商业服务提供商

XML extensible Markup Language 扩展标记语言

SIMD Single Instruction Multiple Data 单指令流多数据流

DCE distributed computing environment 分布式计算环境

URL uniform(统一) resource locator 统一资源定制器

GNU GNU's Not Unix 革奴计划，目标是创建一套完全[自由](http://zh.wikipedia.org/wiki/%E8%87%AA%E7%94%B1%E8%BD%AF%E4%BB%B6)的[操作系统](http://zh.wikipedia.org/wiki/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)

DNS Domain Name System 域名解析系统

UTC universal time coordinated 通用协调时间

CORBA：Common Object Request Broker Architecture 公共对象请求代理体系结构

MPEG：Motion Picture Experts Group 运动图像专家组(全球影象/声音/系统压缩标准)

RMI：Remote Method Invocation 远程方法调用

ISO：International Standardization Organization 国际标准化组织

OSI：Open System Interconnection 开放式系统互联

WAN: wide-area network 广域网络

IDL: interface definition language 接口定义语言

SANs: System Area Networks 系统区域网络

NUMA: Non-uniform memory access 非均衡存储器访问

MPP： Massively Parallel Processors 大规模并行处理器

COWs: cluster of workstations 集群工作站

DSM: distributed shared memory 分布式共享内存

MOM: message-oriented middleware 面向消息的中间件

UDP: universal datagram protocol 通用数据报协议

TCP: Transmission Control Protocol 传输控制协议

RTP: real-time transport protocol 实时传送协议

FTP: file transfer protocol 文件传输协议

HTTP: Hypertext Transfer protocol 超文本传输协议

IPC: interprocess communication

MPI: message-passing interface 消息传递接口

QoS: Quality of Service 服务质量

RSVP: resource reservation protocol 资源预留协议

DIB: directory information base 目录信息库

RDN: relative distinguished name 相对可分辨名称

DIT: directory information tree 目录信息树

DSA: directory service agent 目录服务代理

DUA: directory user agent 目录用户代理

LDAP: light-weight directory access protocol 轻量级目录访问协议

ARP: address resolution protocol 地址解析协议

TAI: international atomic time 国际原子时间

NTP: net time protocol 网络时间协议

ACID: atomic,consistent,isolated,durable 事务的四个特性

2PL: 2-phase locking 2相锁

ROWA: read one write all

TMR: triple modular redundancy 三模冗余

2PC: 2-phase commit protocol

Callback

回调函数，就是由你自己写的。你需要调用另外一个函数，而这个函数的其中一个参数，就是你的这个回调函数名。这样，系统在必要的时候，就会调用你写的回调函数，这样你就可以在回调函数里完成你要做的事。

回调函数可以自动唤醒队列中的信息，也可以启动一个能获取队列中信息的过程。这个回调的过程在接收端进行。

在AFS 服务器会记录客户端的缓存行为，客户端缓存文件的属性和数据。当有其他客户端试图修改已经被别的客户端缓存的文件属性或者数据时，服务器会采用回调（call-back）的机制通知客户端这些缓存已经失效。

Middleware

中间件定义成一个软件层，它的目的是对应用程序隐藏底层平台的异构性，它对应用程序隐藏了实际网络和通信协议的细节，为应用程序员提供方便的编程模型。中间件表示成一组计算机上的进程或对象，他们互相交互，实现分布式应用的通信和资源共享支持。

同步(Synchronization)

多个进程协作或互斥地访问共享资源，多个进程就事件的顺序达成一致。

**二、DNS两种方式的比较：**

迭代名称解析每次解析一个节点以后都会把结果返回给客户名称解析程序，然后将由客户重新向下一级服务器发送请求。

递归名称解析是直接把解析的中间结果传递给下一级服务器，并且逐级解析直到返回指定文件结果。

递归优点：缓存结果有提升，减小了通信负担

递归缺点：要求名称服务器都具有较高性能，增加了额外负担

**三、分布式系统的定义、目标、优缺点、实现方式，结合实例**

分布式系统是一组独立的计算机的集合，它能够使得在用户看来，像一个计算机系统一样

**实现目标**：

1)connect user and resource 能够让用户方便地与资源连接

2)Transparency 隐藏资源在一个网络上分布这样一个事实

3)openness 开放的

4)scalability 可扩展的

**实现方式（答案不确定）**：

硬件（多处理器系统、同构式、异构式）、软件（分布式操作系统、网络操作系统、中间件）



**优点**：

（1）相对独立PC而言：

数据共享：多个用户访问公共的数据库

设备共享：多个用户访问昂贵的设备（激光打印机）

通信：人们之间的通信更加容易（Email）

灵活性：合理的把工作负荷分配到可用的计算机上

（2）相对集中系统：

Economics（经济性）：微处理器能提供比大型机更好的性价比

Speed（速度）：分布式系统能提供比大型机更强的计算能力

Inherent distribution（固有的分布性）：有一些应用包含物理上分布的机器

Reliability（可靠性）：当某台机器崩溃时，整个系统仍能正常工作

Incremental growth（可扩展性）：计算能力逐步增加

**缺点**：

软件：分布式系统开发软件困难

网络：网络可能饱和且可能引起的问题

安全：容易造成对保密数据的访问

**举例**

例如万维网，它向用户提供了一种简单、一致、同意的分布式文档模型。当人们需要浏览一个文档时，只需激活一个到该文档的引用（URL）即可。而且从理论上来讲，用户不需要知道文档来自哪个服务器，更不需要知道该服务器的位置。发布一个文档也很简单，只需对每个要发布的文档提供一个URL即可。

**四、透明性的含义及作用**

对用户和应用程序员屏蔽分布式系统的组件的分散性，系统被认为是一个整体，而不是独立的组件的集合。透明性对用户和应用程序员隐藏了与手头任务无直接关系的资源，并匿名使用。使得分布的某些特性对应用程序员具有不可见性，这样应用程序员只要关心特定应用的设计问题。

（可选）网络透明性：用户察觉不出是以网络的方式工作，所有的工作象是在单台机器上完成。 操作系统透明性：使用不同操作系统的机器和用户可以自由的协同工作，不同操作系统带来的不同被掩盖。 语言透明性：使用不同语言编写的程序或者模块能够自由交互协作，相互调用。 数据复制与分片透明性：分片的数据像未分片的数据一样工作，修改数据时复制在多处的数据自动更新。

分布式透明性实现方式：访问、位置、迁移、重定位、复制、并发、故障、持久性

**五、RPC的目标、过程、优缺点以及动态绑定**

当机器A上的进程调用机器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信息传递过程。这种方法称为远程过程调用（RPC）。

RPC (remote procedure call)过程

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，然后调用本地系统

3、客户端系统将消息发给远程系统

4、远程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，然后调用本地系统

8、服务器系统将含有结果的消息发送回客户端系统

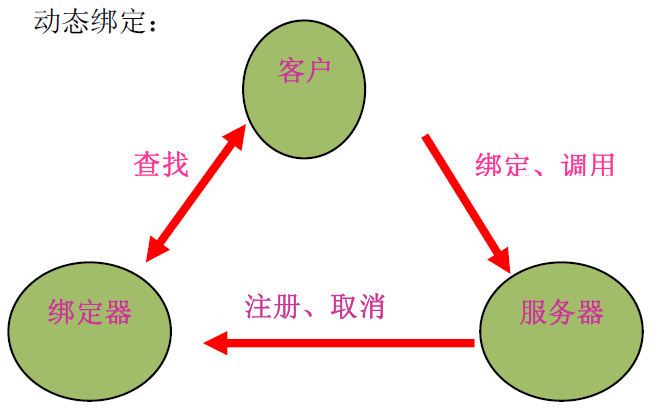
9、客户端系统将消息交给客户存根

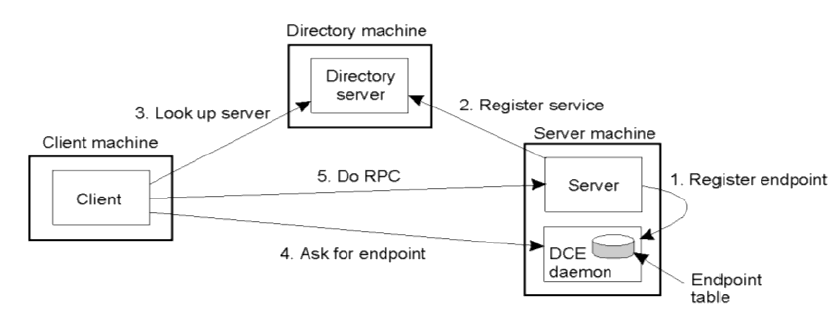
10、客户存根将结果从消息中提取出来，返回给调用它的客户过程

P.S.存根是一段代码

好处：（1）灵活性 （2）处理多服务器支持同一接口的情况 （3）确认服务器和客户是否使用同一版本的接口

缺点：（1）输入输出需要额外的开销 （2）在大型分布式系统中,binder会成为瓶颈。





1Server先向本地OS请求一个端点（endpointt），然后在DCE daemon中注册该端点，记录在端点表中。2Server再向一个目录Server注册自己的网络地址和一个名字。3Client以Server提供的服务名字为key在目录Server中查找Server对应的网络地址，4然后向Server上的DCE daemon请求服务端点。5这些信息都具备后，执行RPC操作。

**六、事务的特性**

事务除了和互斥一样保护共享资源不同时被进程访问以外，还特别允许进程把访问和修改数据项作为一项单独的原子操作来完成。如果进程在事务执行期间退出，那么所有数据都恢复到事务开始之前的状态。所以，事务必须提供对数据库进行恢复的方法。

原子性Atomic：事务是不可发生的瞬间动作，要么全部发生，要么全部不发生，不会存在中间状态；

一致性Consistent：不能破坏系统的恒定性；

独立性Isolated：并发的事务不会相互干扰；

持久性Durable：提交的事务所产生的改变不可恢复，即使提交后出现故障，结果也不可能取消或者丢失。

**七、复制的目的（作用）、困难是什么？数据在复制的时候什么是最大问题？**

对数据进行复制一般是为了提高系统的可靠性或性能。

1. 可靠性。如果一个文件系统已经实现数据复制，那么当一个副本被破坏后，文件系统只需转换到另一个数据副本就可以继续运行下去。
2. 性能。当分布式系统需要在服务器数量和地理区域上进行扩展时，复制对于提高性能是相当重要的。

实现数据复制的一个主要难题是保持各个副本的一致性。简单地说，这个难题就是更新一个拷贝时，必须确保同时更新其他拷贝；否则，数据的各个副本将不再相同。可通过同步操作解决。

**八、一致性模型的作用。关于sequentional consistency、casual consistency、 FIFOconsistency 给了3个事例，让你判断属于那一类(画出结构图)**

一致性模型是分布式数据与进程间的一个“契约”，在该“契约”中每个分布式数据精确的指出了在并发情况下读写操作的结果。

* **严格一致性(strict consistency)：**

对于数据项x的任何读操作将返回最近一次对x进行的写操作的结果所对应的值。严格一致性中存在的问题是它依赖于绝对的全局时间。当数据存储时严格一致性的时候，对于所有的进程来说，所有的写操作都是瞬间可见的，如果执行了读操作，那么无论多快地执行下一个写操作，读操作都将得到当前的值。分布式系统无法实现

* **顺序一致性(sequential consistency)：**

任何执行结果都是相同的，就好像所有进程对数据存储的读、写操作时按照某种序列顺寻执行的，并且每个进程的操作按照程序所指定的顺序出现在这个序列中。

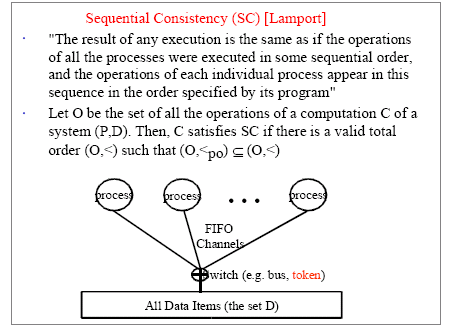
可以把顺序一致性扩展到更严格一些的线性化，就是说虽然假设仍然存在一个全局有效的时间戳，但是精度有限，并且满足顺序一致性的条件。这是就认为数据存储是可线性化的。

因此，线性化和顺序一致性本质上是一样的，只不过后者没有时间戳顺序的要求。

顺序一致性存在严重的性能问题，对于任何的顺序一致性存储，提高读操作性能必将降低写操作性能。



SC: A满足顺序一致性，B不满足



SC结构图

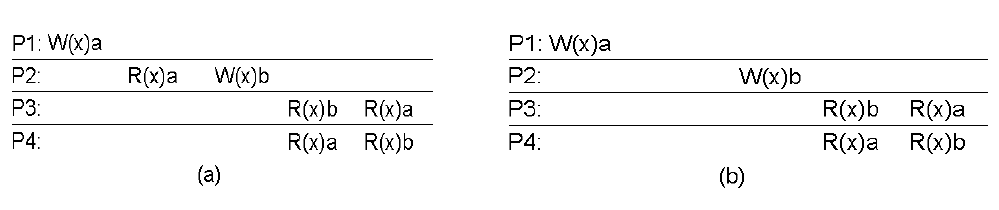
* **因果一致性(causal consistency)**

顺序存储的因果一致性定位为：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被看到并发的写操作。（可以不满足因果一致性）

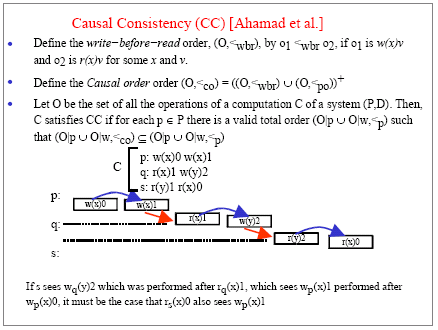
实现因果一致性要求跟踪哪些进程看到了哪些写操作。换句话说，因果一致性允许不同的机器以不同的顺序看到并发的写操作，但是所有的机器必须以相同的顺序看到因果相关的操作。



上图满足因果一致，但不满足顺序和严格一致性



CC: a 不符合（W(x)a与W(x)b有因果关系），b 符合(可以不满足顺序一致性)



CC结构图

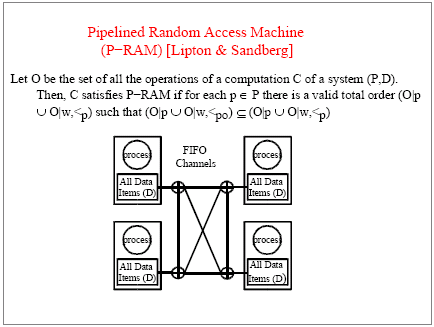
* **FIFO(PRAM)一致性**

所有进程以某个单一进程提出写操作的顺序看到这些写操作，但是不同进程可以以不同的顺序看到不同进程提出的写操作。也就是说，进程不必在开始进行下一个写操作之前停止并且等待前一个写操作的完成。不同进程产生的写操作都是并发的。

FIFO一致性虽然性能上有所改进，但是也存在着一些限制。比如不是所有的应用程序都要求看到所有的写操作，更不会要求按照顺序看到。



FIFO一致性



FIFO结构图

**例题：**

对三个执行历史判断是否满足顺序性、因果和FIFO一致性？

Exe1: p:w(x)1 r(x)2

q:w(x)2 r(x)1

Exe2:p:w(x)1 r(x)2

q:r(x)1 w(x)2

Exe3:p:w(x)3 w(x)1

q:r(x)1 w(y)1

r:r(y)1 r(x)3+

* **弱一致性**

对同步变量的访问必须是顺序一致性的。

在所有前面的写操作完成之前,不能访问同步变量。

在前面所有同步变量的访问完成前,不能访问（读或写）数据。

* **释放一致性**

在访问共享变量前，所有先前的acquire 都必须完成。

在进行release 前，先前的所有读写操作都必须结束。

acquire 和release 访问必须满足处理机（FIFO）一致性

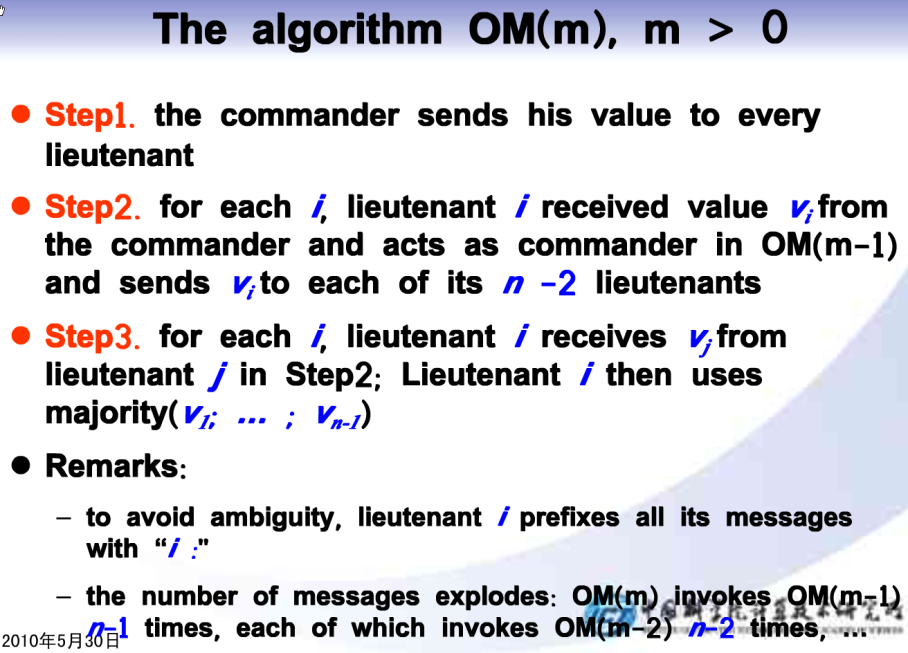
**九、Byzantine Problem**

**算法规则**：

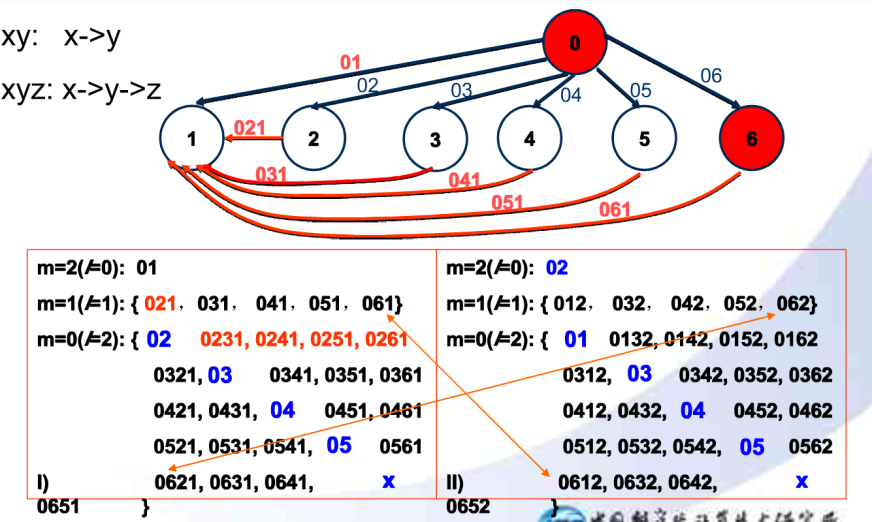
拜占庭将军问题需要满足的两个条件（Interactive Consistency）

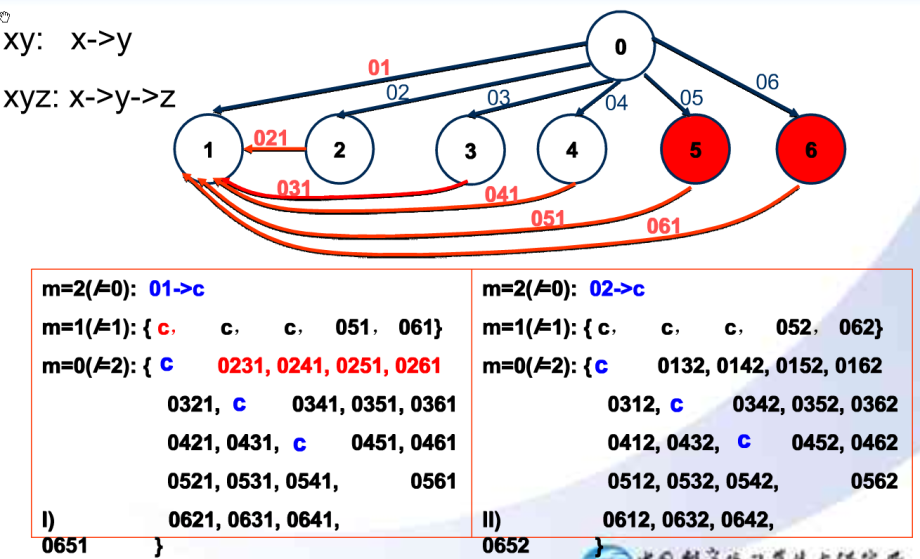
IC1: 所有忠诚的士兵都遵守相同的命令

IC2: 如果指挥官是忠诚的，则所有忠诚的士兵都必须遵守指挥官的命令

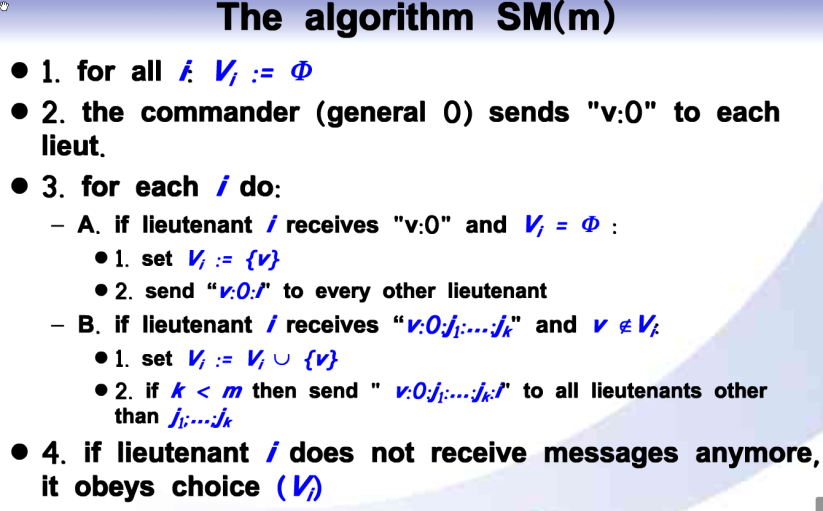


算法规则(终止条件OM(0))

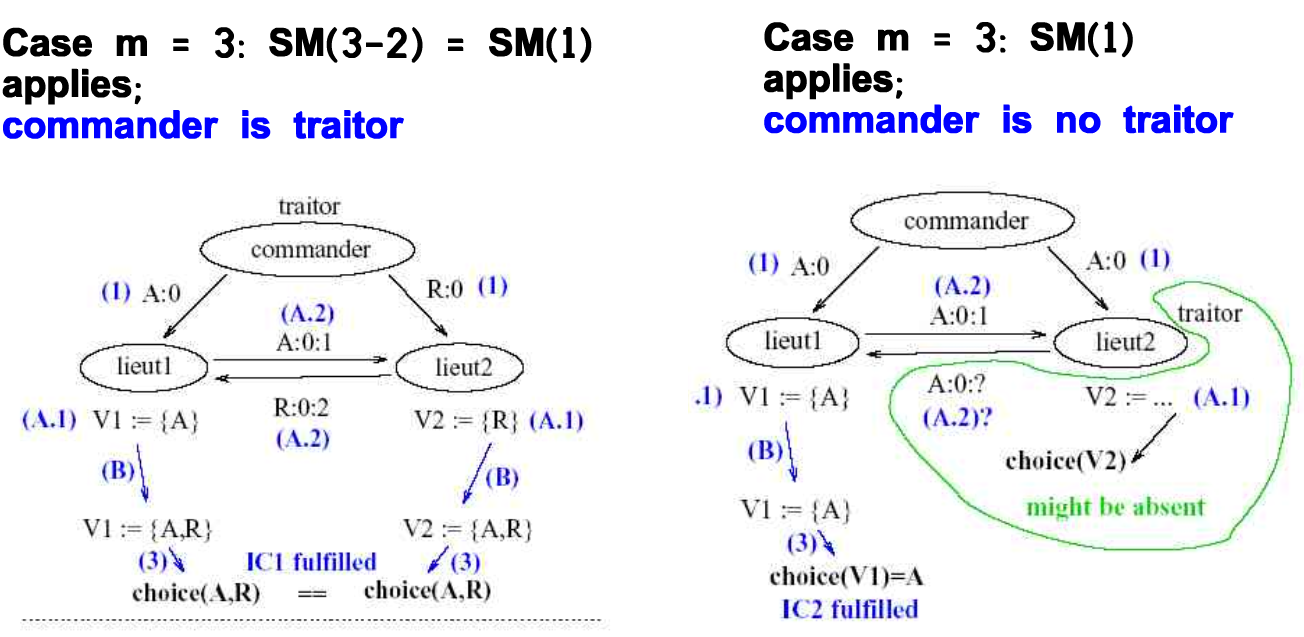




OM(2)算法示例



带签名的规则



带签名的规则示例

1)k-级容错需要多少资源复制

3K+1（Byzantine），非拜占庭情况下2k+1

2)证明m=2,n=6达不成一致（对付n=3m的思路）。

假设他们可以达成一致，则必定满足下列两个条件：

1.所以忠实的中尉服从相同的命令。

2.如果将军是忠实的，则所有忠实的中尉执行将军的命令。

我们现把这6个人分为3组，每两个人一组，假定这样总共有3组，

将军组里有一个将军在原来的6个人的方案里也是将军，其他一个是中尉，

中尉组里都是原来方案中的中尉，这样我们可以看到，分出后的3组仍然是满足上述条件的，于是就得出结论，m=1，n＝3也能达成一直，而从下图中可以看出这是不可能的，所以假设不成立。



3）如果将军对自己所发的信息签名，说明3个节点，1个坏节点可以达成一致；在此情况下猜测，K个坏节点，需要多少个节点 ，经过多少轮信息交换可以达成一致？

解：需要K+2个节点，迭代次数为K轮（如果算上一开始commonder给n-1个节点发送的信息的话，就为K+1轮）

**十、停车场问题**

某停车场共有TOTAL-NUM个车位, ENTRY-NUM个进口, EXIT-NUM个出口. 现需要一个用于停车场控制汽车进出的分布式系统, 在该系统中没有集中的管理者(central server), 每个进(出)口通过通信平等协商保存当前车库的状态信息(如空闲车位数UNOCCUPIED-NUM等), 并据此决定是否允许车辆进入, 为简便计, 假定通信是可靠的.

1) 不考虑节点/进程失效的情形, 设计用于该停车场控制的分布式系统, 并给出汽车进出时使用该系统的方法.

2) 证明你所设计的分布式系统中使用的同步算法满足ME1-ME3\*.

3) 如果新增一个进口节点/进程, 请考虑如何使该进口能参与工作.

1）答：由于几个进口是发布在不同的系统中，所以它们之间对于车位的访问必须是互斥的，所以我采用lamport逻辑时间的方法来实现同步，具体实现如下：

lamport要求先发送的包一定对于接受着来说一定是先收到的，所以下面程序包的发送都采用tcp协议发送。

每个进口都有一个进程号，每个进口的进程号不同。

每个进口里都有一个队列（applicationstack）用来存放消息，该消息包含申请使用临界区资源（车位）的进口号，申请时间。每个进口都有一个整型变量T当该进程每做一个动作就会增加。每个进口有个整型局部变量保存当前空车位，这个变量需要同步访问。当一个进口有车来时，它生成一个类型为application申请时间为当前T的message将这个消息存入自己的applicationstack中，然后T++，然后用当前时间标记这个message向其他进口发送这个application message，每发送一个，T++。当向所有的其他进口发送完消息后它就等待其他所有进口向它回一个reply消息。当收到其他所有进口的reply消息后就检查自己的applicationstack，当发现自己的申请消息是最前的就访问资源。是最前的是指满足下列两个条件

1. 自己的application消息的时间是最小的。

2. 当和别人是并列最小时，自己的进程号时最小的。

到这里这个进口就唯一的获得了临界区资源的访问权，如果空车位数>0则让车进来，否则等待出口释放。

访问完临界资源后，就调用release函数。具体来说，release动作如下：首先将applicationstack中自己的application消息删去，然后向其他所有的进口发送以当前逻辑时间标记的类型为release的消息。到这里一个进口就完全完成了一辆车的进入。

当进口接受到消息后，将自己的逻辑时间T与发送进口发送消息的逻辑时间进行比较，如果是自己大则T++，否则等于T＝发送进口发送消息的逻辑时间＋1，接下来按照消息的类别进行相应的处理，如果是application类型的消息，则把这个消息放入自己applicatonstack中，然后发送一个用当前逻辑时钟标记的reply消息给发送application消息的进口，自己的T++；如果是reply类型的消息则将等待reply数减1，如果是release类型的消息则将在application中的对应与这个发送进口的application类型的消息删去，平且将空车位数减1；如果时car\_out类型的消息，则将空车数加1，并且如果有车在等待进入，则释放等待。

出口只负责当有车离开时向所有的进口发送消息，通知它们空车位有增加，它的动作也很简单，有车出去时，就向所有的进口发送一个当前逻辑时间标识的类型为car\_out的消息。

证明：

3. ME1:只允许有一个进程在临界区中。因为只有当：自己的application消息的时间是最小的或者当和别人是并列最小时，自己的进程号时最小的。一个进口才能获得对临界区资源的访问，每个时间段只有一个进口满足这个条件，并且在这个进口退出之前，其他进口都不满足这个条件。所有同一时间只有一个进程在访问临界区资源。

ME2:对临界区的申请得到满足，每个申请者最后都离开临界区。每个进口当前面的进口使用完临界区资源后都会获得使用临界区的权限，并且在使用完后也会发送release消息释放临界区资源。如果是由于空车位没有在等待，则最终会因为出口的car\_out消息而消除等待，最后释放临界区的使用。

ME3:申请者按照申请顺序获得对临界区资源的访问。

当一个进口在收到别的进口的application消息后，它的T就被修改了，这时它再发送application消息，它applicaton的消息的逻辑时间肯定大于先前向它发送application的逻辑时间，这就表示先申请者先获得临界区资源。如果两个application的逻辑时间是相等的，就说明它们是平发的，这时我们按照进程号大小来安排访问顺序，没有违反ME3。

十一、某分布系统由若干运行在不同结点的进程P1, P2, …, Pn 组成, 考虑各进程的某一变量X1, X2, …, Xn , 要求满足约束 │Xi ━Xj│≤δ. (i, j = 1, 2, …, n). 试设计这样的分布式系统, 并包含一个额外的进程Monitor, 用以监督系统是否满足该约束条件.

解（吕某）：分布式系统概念与设计(原书第3版)中文版 P307

十二、五个哲学家就餐问题，设计一种分布式算法实现。并说明能达到的要求？

解:给哲学家安排优先级和拿叉子的顺序

十三、如何解决消息乱序的问题？

Distributed\_Systems\_Principles\_and\_Paradigms\_2nd\_Edition，vector算法后半部分有解答。此书第一版的解答是错误的。