# 分布式系统

## 作业五

中山大学计算机学院　计算机科学与技术

19335174　施天予

**一、对以下应用程序，你认为最多一次语义和最少一次语义哪个最好？讨论一下。**

(a) 从文件服务器上读写文件;

(b) 编译一个程序;

(c) 远程银行;

(a) 最少一次语义。因为可以重复多次尝试读写文件，失败后继续尝试。

(b) 最少一次语义。同样可以重复多次尝试编译程序。

(c) 最多一次语义。为避免银行资金出现紊乱，服务器保证操作至多执行一次，当出现故障时可以进行人工干预 (比如到银行柜台办理手续) 。

**二、在可靠多播中，通信层是否总是需要重发的目的而保留消息的一个副本？**

不用。比如在传输文件时，只有当数据在应用层可用时以上要求才是必要的。通信层没有必要保留自己的备份。

**三、在两阶段提交协议中，为什么即使在参与者们选择一个新的协作者的情况下也不会完全消除阻塞？**

因为在节点等待消息时处于阻塞状态，其他进程需要等待阻塞进程释放资源才能使用：

- 如果参与者发送同意提交消息给协作者，进程将阻塞直至收到协作者的提交或回滚消息。
- 如果协作者发送“请求提交”消息给参与者，它将被阻塞直到所有参与者回复。

**四、请用例子说明为什么 Paxos 共识算法不满足 liveness？示：参考 PPT 中的例子。**

Paxos 隐含存在活锁的问题，因此不能完全满足 liveness。

比如：Proposer 1 向所有 Acceptors 发送 Prepare 请求，并得到 Acceptors 的肯定答复，这时候新的 Proposer 2 向所有 Acceptors 发送了一个 Proposal ID 更高的 Prepare 请求，这个请求因此会覆盖 Proposer 1 的 Prepare 请求，并得到 Acceptors 的肯定答复。这时候 Proposer 1 并不知道这一点，会继续发送 Accept 请求，然后得到了 Acceptor 的拒绝。于是 Proposer 1 重新开始第一阶段，选取了一个更大的 Proposal ID 来发送 Prepare 请求并得到了 Acceptors 肯定答复。如此往复，一直相互覆盖，难以达成共识。

- $P_1$: prepare($n_1$)
- $P_2$: prepare($n_2$)
- $P_1$: accept($n_1, v_1$)
- $P_2$: accept($n_2, v_2$)
- $P_1$: prepare($n_3$)
- $P_2$: prepare($n_4$)
- ...
- $n_1 < n_2 < n_3 < n_4 < \cdots$

图 1: Paxos 不满足 liveness

**五、Leader selection 在分布式系统中具有重要的用途，主要用于容错，即当主节点失效后能够从备份节点中选择新的 leader，但是新选择的 leader 需要得到其他节点的认同。主流的 leader 选择算法有：Bully、Ring 算法，但不限于这些算法，调研以下软件，简述这些软件所采用的选举算法（任选其二）：Zookeeper、Redis、MongoDB、Cassandra。**

- Zookeeper:

  开始服务器都先给自己投票，每个服务器给自己投票后与其他已投票服务器交换投票信息，id 大的服务器胜出，如果此时投票数过半，则 id 大的服务器成为 leader，否则继续投票，直到出现 id 最大且投票过半的服务器。

- Redis:

  采用哨兵 leader 选举过程，sentinel 会每秒一次向主从和其他 sentinel 发送 ping 来确认 master 是否下线。如果无法收到 master 回应，会认为 master 主观下线，然后向其他 sentinel 确认 master 是否下线，如果收到足够多的消息后将确认 master 客观下线。然后该哨兵会向其他哨兵请求选举自己为 leader，当收到半数的票之后，进行 slave 的选举，选举出来的 slave 会成为 master，向其他的 slave 发送命令使其成为新的 master 的 slave。

- MongoDB:

  不同节点之间进行心跳检查，当没有心跳时会发出选举请求，一轮选举当中每一个成员最多投一票，如果投票时得到超过一半的投票，则会通过选举成为主节点。