**霸道（bully）算法实现**

霸道选举算法是一种分布式选举算法，每次都会选出存活的进程中ID最大的候选者。

**霸道选举算法的选举流程**

**消息类型**

1. 选举消息：表示发起一次选举。
2. 存活消息：对选举消息应答，表明自己的存活状态。
3. 胜利消息：选举胜利者向参与者发送选举成功消息。

**选举触发事件**

1. 进程从错误中恢复；
2. 检测到leader失败。

**选举流程**

（1）如果进程P有最大的ID，直接向所有其它进程发送胜利消息，成功新的Leader；否则向所有比它的ID大的进程发送选举消息；

（2）如果进程P发送选举消息后没有收到存活消息，则它向所有进程发送胜利消息，成功新的Leader；

（3）如果进程P收到比自己ID大的进程发来的存活消息，进程P停止发送任何消息，等待胜利消息（如果过了一段时间没有等到胜利消息，则重新开始选举流程）；

（4）如果进程P收到比自己ID小的进程发来的选举消息，回复一个存活消息，然后重新开始选举流程；

（5）如果进程P收到胜利消息，它把发送者看作Leader。

**霸道选举算法假设**

霸道选举算法的假设包括：

（1）假设通信是可靠的，更进一步的假设系统中任何两个进程之间都可以通信；

（2）假设每个进程都知道其他进程的编号，即算法依赖一个全局的数据。

（3）假设进程能够明确地判断出一个正常运行的进程和一个已经崩溃的进程。

选项一：实现多主机霸道选举算法程序，并测试算法的有效性；

选项二：使用多进程模拟霸道选举算法，并测试算法的有效性。

**选项一：实现多主机霸道选举算法程序，并测试算法的有效性**

1. 参与选举算法的主机及端口等相关信息写入一个配置文件，此配置文件在每台主机启动时首先读取该配置文件。
2. 每台主机上应用启动时，读取配置文件，发起选举过程；
3. 定义通信过程中使用的消息的消息报文格式及数据结构，除了协议中提到的三类消息，可根据需要自定义使用到的其它消息及消息报文格式。
4. 处理两类选举触发事件；
5. 测试（局域网测试，4-5台机器）
   1. 测试各台机器同时启动时发起的选举过程，通过日志绘制各台机器间选举报文交换情况；
   2. 测试某一台机器宕机后选举过程，通过日志绘制各台机器间选举报文交换情况；
   3. 测试某一台机器宕机后重启，通过日志绘制各台机器间选举报文交换情况；

具体实现可以仔细参考论文【1】及附录

选项二：使用多进程模拟霸道选举算法，并测试算法的有效性。

1. 利用进程来模拟主机；
2. 用进程间通信如队列、管道机制来实现模拟主机间通信；
3. 定义各类消息格式；
4. 用睡眠、超时等机制模拟主机宕机状况的发生；
5. 实现进程间选举算法，处理两类选举触发事件
6. 测试。

具体实现可以仔细参考论文【1】及附录

参考文献

1. <http://vis.usal.es/rodrigo/documentos/papers/BullyAlgorithm.pdf>
2. <http://www.cs.colostate.edu/~cs551/CourseNotes/Synchronization/BullyExample.html>
3. [https://en.wikipedia .org/wiki/Bully\_algorithm](https://en.wikipedia.org/wiki/Bully_algorithm)
4. <http://www.xatlantis.ch/index.php/projects/zeus-framework/38-rmi-for-c>
5. <http://bitsavers.org/pdf/xerox/parc/techReports/CSL-81-9_Remote_Procedure_Call.pdf>
6. <http://www.drdobbs.com/implementing-rmi-for-c-objects/184403949>