4 算法实现

对于一个节点来说，其包含proposer, accepter和learner三个角色. 为便于实现，我们分别编写了proposer, accepter和learner. 由于paxos算法主要集中于proposer和accepter的交互，所以我们这里不再关心learner的输出。

代码文件结构如下：

|  |
| --- |
| paxos-main  ├─ acceptor.go //模拟accepter  ├─ go.mod  ├─ learner.go //模拟learner  ├─ messager.go //定义RPC消息结构  ├─ output.txt //代码输出结果  ├─ proposer.go //模拟proposer  └─ test\_test.go //执行和验证paxos |

4.1 核心代码

Proposer.go

（1）实现两阶段协议：

a. phase1，proposer向accepter发送prepare消息；如果得到了半数以上的accepter的promise，则该proposer可以准备发送accept请求消息。否则，该proposer重新提议。

b. phase2，proposer向得到accepter发送accept请求消息；如果得到了半数以上的accepter的accepted消息，则该proposer的提议达成了共识，该轮paxos结束。

|  |
| --- |
| **func** **(**p **\***Proposer**)** runTwoPhase**()** **{**  peers **:=** p**.**getAcceptorPeers**()**  **for** p**.**decidedValue **==** **nil** **{**  // phase 1  prepareMsgReq **:=** p**.**prepare**()**  **var** promiseSuccessNum **int64**  **var** promiseFailedNum **int64**  promiseSuccessChan **:=** **make(chan** **struct{})**  promiseFailedChan **:=** **make(chan** **struct{})**  logPrint**(**"[proposer:%d] phase 1, prepareMsg:%v"**,** p**.**getMe**(),** prepareMsgReq**)**  **for** **\_,** peerAddr **:=** **range** peers **{**  **go** **func(**peerAddr **string,** prepareMsgReq PrepareMsg**)** **{**  **defer** **func()** **{**  **if** atomic**.**LoadInt64**(&**promiseSuccessNum**)** **>=** p**.**getQuorumSize**()** **{**  promiseSuccessChan **<-** **struct{}{}**  **return**  **}**  **if** atomic**.**LoadInt64**(&**promiseFailedNum**)** **>=** p**.**getQuorumSize**()** **{**  promiseFailedChan **<-** **struct{}{}**  **return**  **}**  **}()**  promiseMsgResp**,** err **:=** p**.**sendPrepare**(**peerAddr**,** **&**prepareMsgReq**)** // 这里用同步的方式处理，就不考虑响应消息重复接收了  **if** err **!=** **nil** **||** prepareMsgReq**.**ProposeID **!=** promiseMsgResp**.**ProposeID **||**  promiseMsgResp**.**AcceptorAddr **!=** peerAddr **{**  atomic**.**AddInt64**(&**promiseFailedNum**,** 1**)**  **return**  **}**  **if** promiseMsgResp**.**Success **{**  atomic**.**AddInt64**(&**promiseSuccessNum**,** 1**)**  **}** **else** **{**  atomic**.**AddInt64**(&**promiseFailedNum**,** 1**)**  **}**  **if** promiseMsgResp**.**AccepedID **>** 0 **{**  p**.**setAccepted**(**promiseMsgResp**)**  **}**  **}(**peerAddr**,** prepareMsgReq**)**  **}**  **select** **{**  **case** **<-**promiseFailedChan**:**  logPrint**(**"[proposer:%d] %v 小于当前accepter的N"**,** p**.**getMe**(),** p**.**proposerID**)**  **continue**  **case** **<-**promiseSuccessChan**:**  logPrint**(**"[proposer:%d] %v 有可能会达到共识"**,** p**.**getMe**(),** p**.**proposerID**)**//当前pomise已到半数  // prepare success  **}**  // phase 2  **var** acceptSuccessNum **int64**  **var** acceptFailedNum **int64**  acceptSuccesChan **:=** **make(chan** **struct{})**  acceptFailedChan **:=** **make(chan** **struct{})**  acceptMsgReq **:=** p**.**accept**()**  logPrint**(**"[proposer:%d] phase 2, acceptMsg:%v(最后一个值是proposer计划提案值)"**,** p**.**getMe**(),** acceptMsgReq**)**  **for** **\_,** peerAddr **:=** **range** peers **{**  **go** **func(**peerAddr **string,** acceptMsgReq AcceptMsg**)** **{**  **defer** **func()** **{**  **if** atomic**.**LoadInt64**(&**acceptSuccessNum**)** **>=** p**.**getQuorumSize**()** **{**  acceptSuccesChan **<-** **struct{}{}**  **return**  **}**  **if** atomic**.**LoadInt64**(&**acceptFailedNum**)** **>=** p**.**getQuorumSize**()** **{**  acceptFailedChan **<-** **struct{}{}**  **return**  **}**  **}()**  acceptedMsgResp**,** err **:=** p**.**sendAccept**(**peerAddr**,** **&**acceptMsgReq**)** // 这里用同步的方式处理，就不考虑响应消息重复接收了  **if** err **!=** **nil** **||** acceptMsgReq**.**ProposeID **!=** acceptedMsgResp**.**ProposeID **||**  peerAddr **!=** acceptedMsgResp**.**AcceptorAddr **{**  atomic**.**AddInt64**(&**acceptFailedNum**,** 1**)**  **return**  **}**  **if** acceptedMsgResp**.**Success **{**  atomic**.**AddInt64**(&**acceptSuccessNum**,** 1**)**  **}** **else** **{**  atomic**.**AddInt64**(&**acceptFailedNum**,** 1**)**  **}**  **}(**peerAddr**,** acceptMsgReq**)**  **}**  **select** **{**  **case** **<-**acceptFailedChan**:**  logPrint**(**"[proposer:%d] %v not reach consensuse"**,** p**.**getMe**(),** p**.**proposerID**)**  **continue**  **case** **<-**acceptSuccesChan**:**  p**.**decidedValue **=** acceptMsgReq**.**Value  logPrint**(**"[proposer:%d] %v reach consensuse Value: %v"**,** p**.**getMe**(),** p**.**proposerID**,** acceptMsgReq**.**Value**)**  **return**  **}**  **}**  **}** |

Accepter.go

（1）接收proposer的prepare消息和向proposer发送promise消息。

每个accepter维护一个max\_n（该accepter回复promise的最大提议号），如果发送prepare的提议号n>max\_n，则max\_n=n，accepter向发送该prepare的proposer发送promise（承诺不再接受比n更小的提议号）。如果accepter曾经accept过提议内容，则返回的promise消息出了包括此次提议号以外，还会包括曾经accept过的内容和对应的提议号。

|  |
| --- |
| **func** **(**a **\***Acceptor**)** RecievePrepare**(**arg **\***PrepareMsg**,** reply **\***PromiseMsg**)** error **{**  logPrint**(**"[acceptor %s RecievePrepare:%v ]"**,** a**.**localAddr**,** arg**.**ProposeID**)**  reply**.**ProposeID **=** arg**.**ProposeID  reply**.**AcceptorAddr **=** a**.**getAddr**()**  **if** arg**.**ProposeID **>** a**.**promiseID **{**  a**.**promiseID **=** arg**.**ProposeID  reply**.**Success **=** **true**  **var** flag **=false**  **if** a**.**acceptedID **>** 0 **&&** a**.**acceptedValue **!=** **nil** **{**  flag **=true**  logPrint**(**"[acceptor %s SendPromise:%v,(%v,%v) ]"**,** a**.**localAddr**,** a**.**promiseID**,**a**.**acceptedID**,**a**.**acceptedValue**)**//add  reply**.**AccepedID **=** a**.**acceptedID  reply**.**AccepedValue **=** a**.**acceptedValue  **}**  **if** **!**flag **{**  logPrint**(**"[acceptor %s SendPromise:%v ]"**,** a**.**localAddr**,** a**.**promiseID**)**//add  **}**    **}** |

（2）接收proposer的accept请求消息和向proposer发送accepted消息。

如果accepter曾经promise过accept请求消息中的提议号，且没有再promise或者accept过更高的提议号，这时候accepter会向此次提议回复accepted消息。

|  |
| --- |
| **func** **(**a **\***Acceptor**)** RecieveAccept**(**arg **\***AcceptMsg**,** reply **\***AcceptedMsg**)** error **{**  logPrint**(**"[acceptor %s RecieveAccept:%v ]"**,** a**.**localAddr**,** arg**.**ProposeID**)**  reply**.**ProposeID **=** arg**.**ProposeID  **if** arg**.**ProposeID **==** a**.**promiseID **{**  logPrint**(**"[acceptor %s SendAccept:%v ]"**,** a**.**localAddr**,** arg**.**ProposeID**)**//add  reply**.**Success **=** **true**  reply**.**AcceptorAddr **=** a**.**getAddr**()**  a**.**promiseID **=** arg**.**ProposeID  a**.**acceptedID **=** arg**.**ProposeID  a**.**acceptedValue **=** arg**.**Value  **for** **\_,** learnerPeer **:=** **range** a**.**getLearnerPeers**()** **{**  callRpc**(**learnerPeer**,** "Learner"**,** "RecieveAccepted"**,** reply**,** **&**EmptyMsg**{})**  **}**  **}**  // PASS 持久化accepted的数据  **return** **nil**  **}** |

Messager.go

（1）提议号的标识使用’n.p’的形式，p代表proposer的编号，n代表p第几次提议。如’1.1’代表第一个proposer第一次提议的提议号。P随proposer变化，n单调递增。

|  |
| --- |
| **func** generateNumber**(**me **int,** number **float32)** **float32** **{**  **var** strNum **string**  **if** number **==** 0 **{**  strNum **=** fmt**.**Sprintf**(**"1.%d"**,** me**)**  n**,** err **:=** strconv**.**ParseFloat**(**strNum**,** 32**)**  **if** err **!=** **nil** **{**  **panic(**"error parse num"**)**  **}**  **return** **float32(**n**)**  **}**  i **:=** **int(**number**)** **+** 1 // 暂时不考虑float的精度问题  strNum **=** fmt**.**Sprintf**(**"%d.%d"**,** i**,** me**)**  n**,** err **:=** strconv**.**ParseFloat**(**strNum**,** 32**)**  **if** err **!=** **nil** **{**  **panic(**"error parse num"**)**  **}**  **return** **float32(**n**)**  **}** |

（2）定义phase1的消息：proposer发送的prepare消息和accepter发送的promise消息。

|  |
| --- |
| **type** PrepareMsg **struct** **{**  ProposeID **float32**  **}**  **type** PromiseMsg **struct** **{**  AcceptorAddr **string**  ProposeID **float32**  Success **bool**  AccepedID **float32**  AccepedValue **interface{}**  **}** |

（3）定义phase2的消息：proposer发送的accept消息定义accepter发送的accepted消息。

|  |
| --- |
| **type** AcceptMsg **struct** **{**  ProposeID **float32**  AcceptorAddr **string**  Value **interface{}**  Value0 **interface{}**  **}**  **type** AcceptedMsg **struct** **{**  ProposeID **float32**  AcceptorAddr **string**  Success **bool**  **}** |

Test\_test.go

建立集群，初始化定义的proposer，accepter和learner. Proposer可以以编号代表，proposer 1代表第一个提议者。127.0.0.1:4101代表第1个accepter，127.0.0.1:4102代表第二个accepter，依次类推。

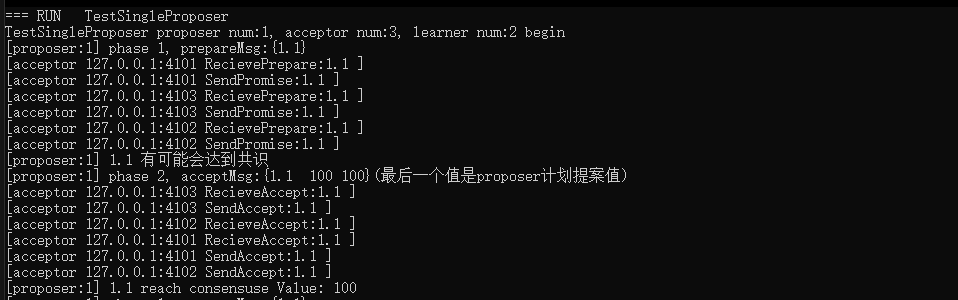
|  |
| --- |
| // 初始化集群  **func** makeCluster**(**proposerNum**,** acceptorNum**,** learnNum **int)** **(**p **[]\***Proposer**,** a **[]\***Acceptor**,** l **[]\***Learner**)** **{**  acceptorPort **:=** 4100  learnerPort **:=** 4200  learnPeers **:=** **make([]string,** learnNum**)**  **for** i **:=** 0; i **<** learnNum; i**++** **{**  learnerPort**++**  learner **:=** **&**Learner**{**  localAddr**:** formatAddr**(**learnerPort**),**  **}**  l **=** **append(**l**,** learner**)**  learnPeers**[**i**]** **=** learner**.**localAddr  l**[**i**].**startRpc**()**  **}**  acceptorPeers **:=** **make([]string,** acceptorNum**)**  **for** i **:=** 0; i **<** acceptorNum; i**++** **{**  acceptorPort**++**  acceptor **:=** **&**Acceptor**{**  localAddr**:** formatAddr**(**acceptorPort**),**  learnerPeers**:** learnPeers**,**  **}**  a **=** **append(**a**,** acceptor**)**  acceptorPeers**[**i**]** **=** acceptor**.**localAddr  a**[**i**].**startRpc**()**  **}**  **for** i **:=** 0; i **<** proposerNum; i**++** **{**  proposer **:=** **&**Proposer**{**  acceptorPeers**:** acceptorPeers**,**  me**:** i **+** 1**,**  **}**  p **=** **append(**p**,** proposer**)**  **}**  **return**  **}** |

4.2 实验结果

代码全部运行结果见文件output.txt.下面展示三种情况下，basic paxos的实验验证结果。

4.2.1 单proposer，多accepter（accepter可靠）

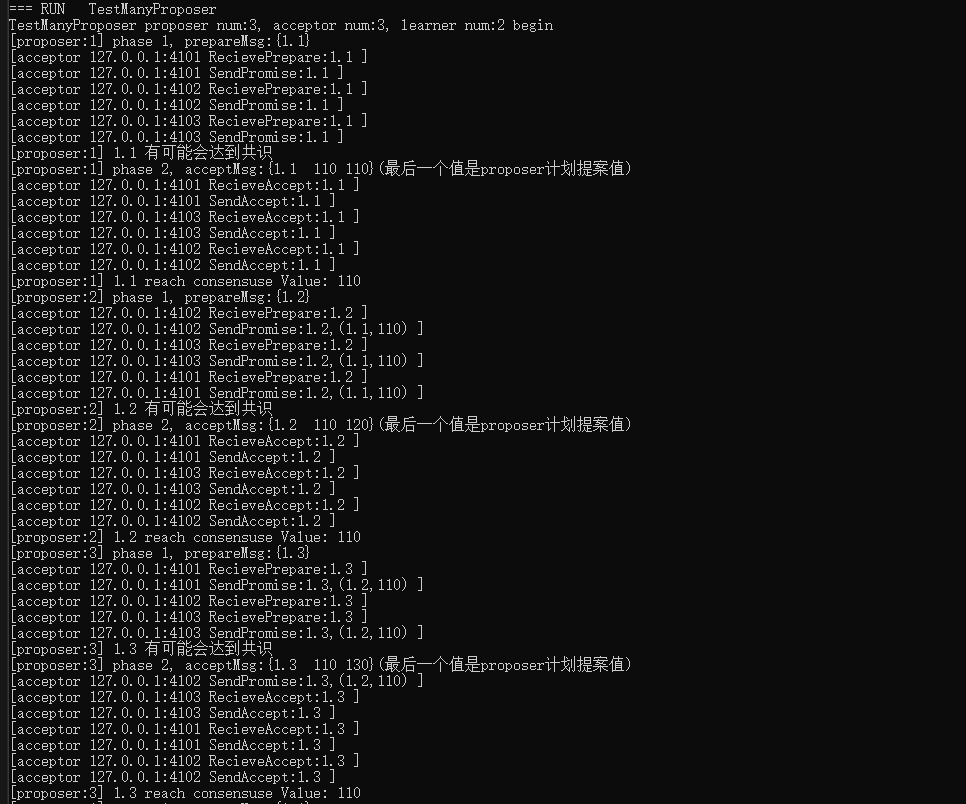
实验中设置proposer个数为1，accepter个数为3.



Proposer1向accepter发送prepare请求，3个accepter（4101, 4102, 4103）都收到了prepare请求也也都向proposer发送promise。Proposer1收到了大于半数的promise（这个时候提议1.1已经有可能达成共识了），准备发送accept请求。3个accepter也都收到了accept请求并并接受了accept请求（向proposer发送accepted）。

4.2.2 多proposer，多accepter（accepter可靠）

设置3个proposer和3个accepter.

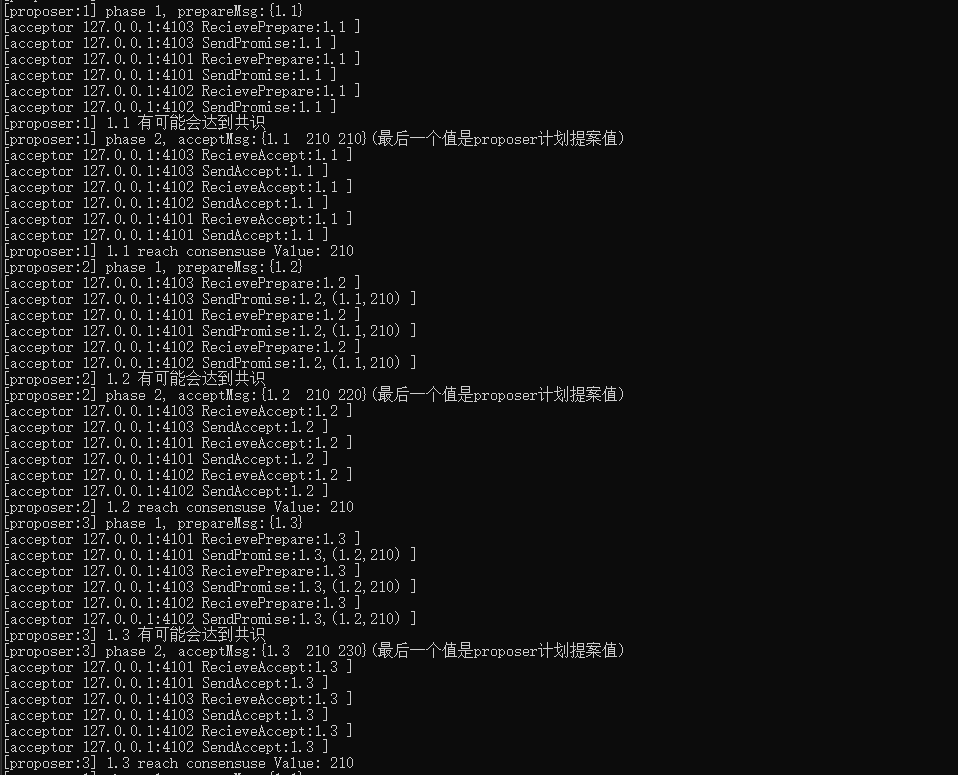


和前文介绍’n.p’表示一致，这里’1.1’代表proposer1第一次提议的提议号；’1.2’代表proposer2第一次提议号。

对于proposer1的提议过程和和前面单proposer一致。

对于proposer2，向accepter发送prepare请求时，也收到了3个accepter的promise. 与proposer1不同的是，proposer2原本要提议的内容是120，而当3个accepter曾经对proposer1的提议号为1.1的提议内容达成了共识，所以，返回给proposer2的promise还会包括(1.1, 110)（三个accepter曾经接受过的最大的提议号和提议号对应的提议内容）。

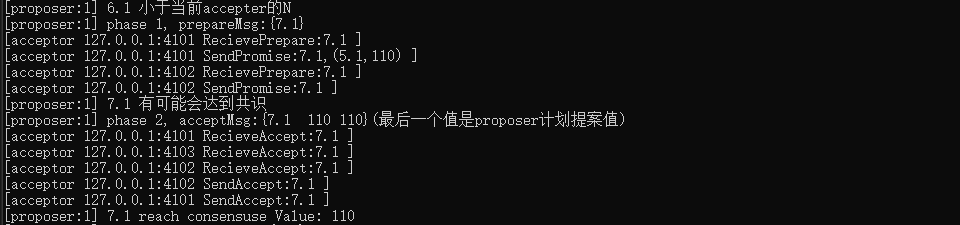
如果要达成多个共识，可以多轮运行paxos算法，如下面的结果，2.1提议内容为210：



4.2.3 多proposer，多accepter（accepter不可靠）

Accepter不可靠时，随时可能“离线”，不能及时接收到proposer的请求，因为也不能成功回复proposer.

（1）达成共识（全员accepter达成共识，大于半数accepter达成共识）



6.1的提议没有达到共识，7.1开始。7.1收到了accepter1，2的promise和accept（发送accept的accepter数量为2大于accepter总数3的一半），所以7.1达成共识。（由于第3个accepter没有发送过promise，所以不会向7.1发送accept）。

（2）未达成共识（收到了少于半数的accept）

a. 在phase1失败.

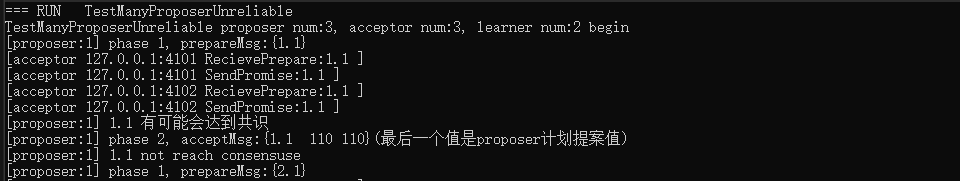


Proposer1第二次提议时没有收到大于半数的accepter的promise，也就没有可能继续收到超过半数的accepter的accept, 所以增大提议号，开始新一轮（3.1）。

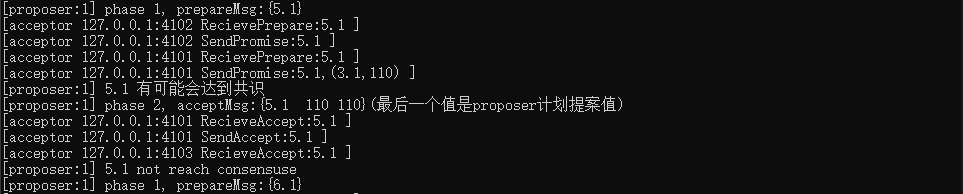


由于accepter不可靠，5.3没有收到promise，开始新的一轮（6.3）。

b. 在phase2失败

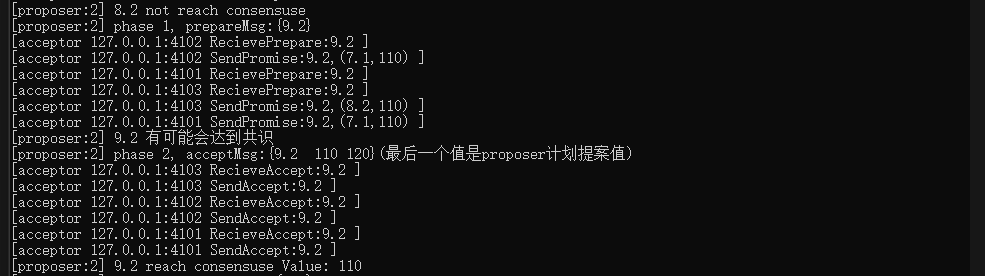


1.1收到了accepter1和accepter2的promise，是有可能达成共识的。1.1开始发送accept请求，但后面由于没有收到accepter的accept, 所以最终1.1没有达成共识。于是增大提议号，开始新一轮的请求(2.1)。

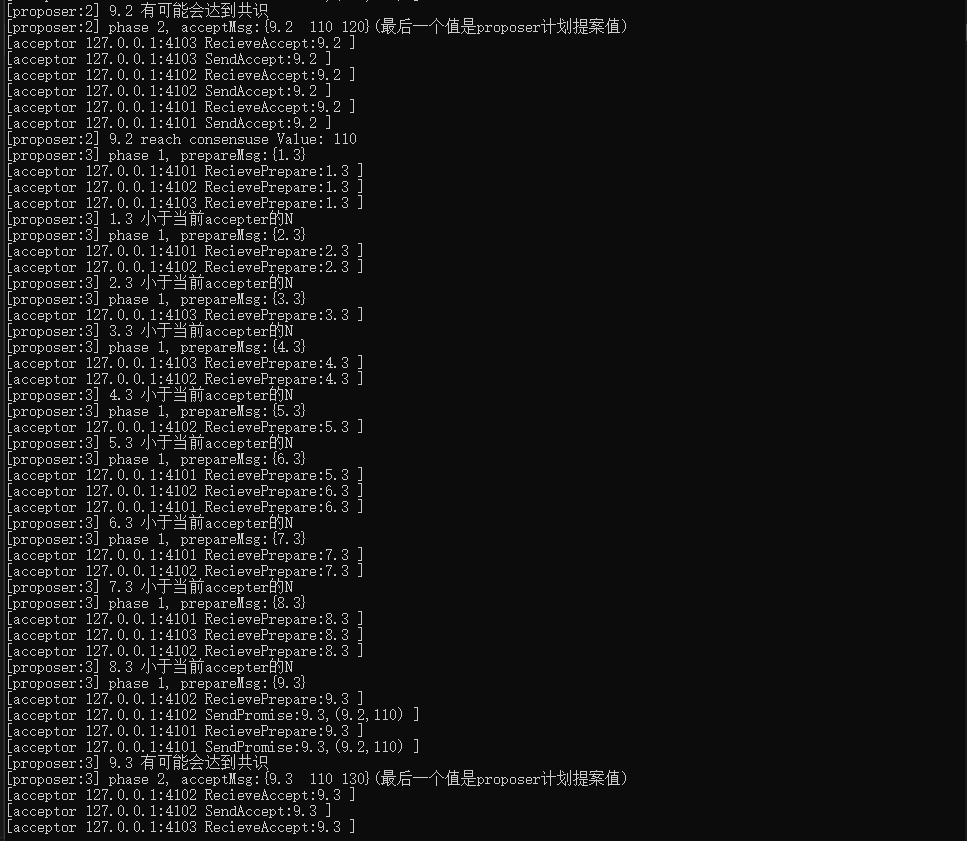


5.1收到了超过半数的promise（来自accepter1和accepter2的回复），但没有收到accepter的accept请求，所以此次提议失败，增大提议号为6.1开始新一轮提议。（因为第3个accepter没有向提议5.1发送给promise，所以也不会向5.1发送accept）。

如果proposer一直没有达成共识，会一直开始新的提议。（如下图，第二个proposer在第9次提议才达到共识）



（3）情况补充



因为3个accepter曾经都接受过9.2的提议，对于3个accepter本地维护的max\_n此时是9.2. 当第三个proposer提议时，如果提议号小于9.2则不会得到accepter的响应。