PBFT 算法实验指导书

实验课时：3 学时
实验认识 3-4 人

一、实验名称

PBFT 共识算法的实践

二、实验内容

理解 PBFT 算法的工作原理和流程，能够使用编程语言实现简单的 PBFT 算法。

三、实验环境

GO 开发环境：go1.20.4

IDE 工具：vscode

操作系统：windows11

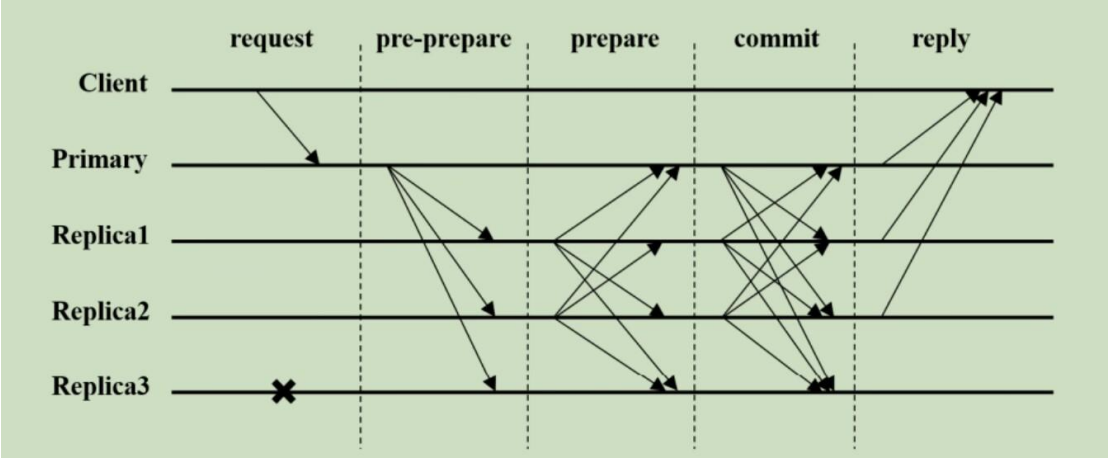
四、算法描述

PBFT(Practical Byzantine Fault Tolerance) 是一种分布式系统中的一致性算法，用于解决拜占庭将军问题。PBFT 算法是基于拜占庭容错性的，它可以在有多个节点的情况下，保证数据的一致性。

PBFT 算法的消息传递过程包括四个阶段：准备阶段、传播阶段、确认阶段和终止阶段。在准备阶段，节点准备好消息并等待其他节点响应。在传播阶段，节点将消息传递给其他节点。在确认阶段，收到消息的节点向其他节点发送确认消息。在终止阶段，所有节点完成任务并终止。

五、实验过程

PBFT 算法的流程如下：



数据结构：

```
// 本地消息池（模拟持久化层），只有确认提交成功后才会存入此池
var localMessagePool = []Message{}

type node struct {
    //节点 ID
    nodeID string
    //节点监听地址
    addr string
    //RSA 私钥
    rsaPrivKey []byte
    //RSA 公钥
```

```
    rsaPubKey []byte
}
type pbft struct {
    //节点信息
    node node
    //每笔请求自增序号
    sequenceID int
    //锁
    lock sync.Mutex
    //临时消息池，消息摘要对应消息本体
    messagePool map[string]Request
    //存放收到的 prepare 数量(至少需要收到并确认 2f 个)，根据摘要来对应
    prePareConfirmCount map[string]map[string]bool
    //存放收到的 commit 数量(至少需要收到并确认 2f+1 个)，根据摘要来对应
    commitConfirmCount map[string]map[string]bool
    //该笔消息是否已进行 Commit 广播
    isCommitBordcast map[string]bool
    //该笔消息是否已对客户端进行 Reply
    isReply map[string]bool
}
```

1. request 阶段:客户端向主节点发送请求信息。

```
func (p *pbft) handleClientRequest(content []byte) {
    fmt.Println("主节点已接收到客户端发来的 request ...")
    //使用 json 解析出 Request 结构体
    r := new(Request)
    err := json.Unmarshal(content, r)
    if err != nil {
        log.Panic(err)
    }
    //添加信息序号
    p.sequenceIDAdd()
    //获取消息摘要
    digest := getDigest(*r)
    fmt.Println("收到的 request 消息为: ", r.Message)
    fmt.Println("已将 request 存入临时消息池")
    //存入临时消息池
    p.messagePool[digest] = *r
    //主节点对消息摘要进行签名
    digestByte, _ := hex.DecodeString(digest)
    signInfo := p.RsaSignWithSha256(digestByte, p.node.rsaPrivKey)
    //拼接成 PrePrepare，准备发往 follower 节点
    pp := PrePrepare{*r, digest, p.sequenceID, signInfo}
    b, err := json.Marshal(pp)
    if err != nil {
```

```
        log.Panic(err)
    }
    pause()
    fmt.Println("正在向其他节点进行进行 PrePrepare 广播 ...")
    fmt.Println("PrePrepare 消息内容为: ", pp)
    //进行 PrePrepare 广播
    p.broadcast(cPrePrepare, b)
    fmt.Println("PrePrepare 广播完成")
    pause()
}
```

2. pre-prepare 阶段：主节点 N0 接收到客户端请求后将请求数据里的主要信息提出，并向其余节点进行 preprepare 发送。

```
func (p *pbft) handlePrePrepare(content []byte) {
    fmt.Println("本节点已接收到主节点发来的 PrePrepare ...")
    pause()
    //  //使用 json 解析出 PrePrepare 结构体
    pp := new(PrePrepare)
    err := json.Unmarshal(content, pp)
    if err != nil {
        log.Panic(err)
    }
    //获取主节点的公钥，用于数字签名验证
    primaryNodePubKey := p.getPubKey("N0")
    digestByte, _ := hex.DecodeString(pp.Digest)
    if digest := getDigest(pp.RequestMessage); digest != pp.Digest {
        fmt.Println("信息摘要对不上，拒绝进行 prepare 广播")
    } else if p.sequenceID+1 != pp.SequenceID {
        fmt.Println("消息序号对不上，拒绝进行 prepare 广播")
    } else if !p.RsaVerySignWithSha256(digestByte, pp.Sign, primaryNodePubKey) {
        fmt.Println("主节点签名验证失败！,拒绝进行 prepare 广播")
    } else {
        //序号赋值
        p.sequenceID = pp.SequenceID
        //将信息存入临时消息池
        fmt.Println("已将消息存入临时节点池")
        p.messagePool[pp.Digest] = pp.RequestMessage
        //节点使用私钥对其签名
        sign := p.RsaSignWithSha256(digestByte, p.node.rsaPrivKey)
        //拼接成 Prepare
        pre := Prepare{pp.Digest, pp.SequenceID, p.node.nodeID, sign}
        bPre, err := json.Marshal(pre)
        if err != nil {
            log.Panic(err)
```

```
        }
        //进行准备阶段的广播
        fmt.Println("正在进行 Prepare 广播 ...")
        fmt.Println("广播的 Prepare 消息内容为: ", pre)
        p.broadcast(cPrepare, bPre)
        fmt.Println("Prepare 广播完成")
        reader := bufio.NewReader(os.Stdin)
        fmt.Print("Press enter to continue...")
        _, _ = reader.ReadString('\n')
    }
}
```

3. prepare 阶段：从节点们接收到来自主节点的 preprepare，首先利用主节点的公钥进行签名认证，其次将消息进行散列（消息摘要，以便缩小信息在网络中的传输大小）后，向其他节点广播 prepare。

```
func (p *pbft) handlePrepare(content []byte) {
    //使用 json 解析出 Prepare 结构体
    pre := new(Prepare)
    err := json.Unmarshal(content, pre)
    if err != nil {
        log.Panic(err)
    }
    fmt.Printf("本节点已接收到%s 节点发来的 Prepare ... \n", pre.NodeID)
    //获取消息源节点的公钥，用于数字签名验证
    MessageNodePubKey := p.getPubKey(pre.NodeID)
    digestByte, _ := hex.DecodeString(pre.Digest)
    if _, ok := p.messagePool[pre.Digest]; !ok {
        fmt.Println("当前临时消息池无此摘要，拒绝执行 commit 广播")
    } else if p.sequenceID != pre.SequenceID {
        fmt.Println("消息序号对不上，拒绝执行 commit 广播")
    } else if !p.RsaVerySignWithSha256(digestByte, pre.Sign, MessageNodePubKey) {
        fmt.Println("节点签名验证失败！,拒绝执行 commit 广播")
    } else {
        p.setPrePareConfirmMap(pre.Digest, pre.NodeID, true)
        count := 0
        for range p.prePareConfirmCount[pre.Digest] {
            count++
        }
        //因为主节点不会发送 Prepare，所以不包含自己
        specifiedCount := 0
        if p.node.nodeID == "N0" {
            specifiedCount = nodeCount / 3 * 2
        } else {
```

```
            specifiedCount = (nodeCount / 3 * 2) - 1
        }
        //如果节点至少收到了 2f 个 prepare 的消息（包括自己）,并且没有进行过 commit 广播，则进行 commit
广播
        p.lock.Lock()
        //获取消息源节点的公钥，用于数字签名验证
        if count >= specifiedCount && !p.isCommitBordcast[pre.Digest] {
            pause()
            fmt.Println("本节点已收到至少 2f 个节点(包括本地节点)发来的 Prepare 信息，内容为： ", pre)
            //节点使用私钥对其签名
            sign := p.RsaSignWithSha256(digestByte, p.node.rsaPrivKey)
            c := Commit{pre.Digest, pre.SequenceID, p.node.nodeID, sign}
            bc, err := json.Marshal(c)
            if err != nil {
                log.Panic(err)
            }
            //进行提交信息的广播
            fmt.Println("正在进行 commit 广播 ...")
            fmt.Println("广播的 commit 消息内容为: ", bc)
            p.broadcast(cCommit, bc)
            p.isCommitBordcast[pre.Digest] = true
            fmt.Println("commit 广播完成")
        }
        p.lock.Unlock()
        pause()
    }
}
```

4. commit 阶段：节点接收到 2f 个 prepare 信息（包含自己）,并全部签名验证通过，则可以进行到 commit 步骤，向全网其他节点广播 commit。

```
func (p *pbft) handleCommit(content []byte) {
    //使用 json 解析出 Commit 结构体
    c := new(Commit)
    err := json.Unmarshal(content, c)
    if err != nil {
        log.Panic(err)
    }
    fmt.Printf("本节点已接收到%s 节点发来的 Commit ... \n", c.NodeID)
    //获取消息源节点的公钥，用于数字签名验证
    MessageNodePubKey := p.getPubKey(c.NodeID)
    digestByte, _ := hex.DecodeString(c.Digest)
    if _, ok := p.prePareConfirmCount[c.Digest]; !ok {
        fmt.Println("当前 prepare 池无此摘要，拒绝将信息持久化到本地消息池")
    } else if p.sequenceID != c.SequenceID {
```

```
        fmt.Println("消息序号对不上，拒绝将信息持久化到本地消息池")
    } else if !p.RsaVerySignWithSha256(digestByte, c.Sign, MessageNodePubKey) {
        fmt.Println("节点签名验证失败！,拒绝将信息持久化到本地消息池")
    } else {
        p.setCommitConfirmMap(c.Digest, c.NodeID, true)
        count := 0
        for range p.commitConfirmCount[c.Digest] {
            count++
        }
        //如果节点至少收到了 2f+1 个 commit 消息（包括自己）,并且节点没有回复过,并且已进行过 commit 广播,
则提交信息至本地消息池，并 reply 成功标志至客户端！
        p.lock.Lock()
        if count >= nodeCount/3*2 && !p.isReply[c.Digest] && p.isCommitBordcast[c.Digest] {
            fmt.Println("本节点已收到至少 2f + 1 个节点(包括本地节点)发来的 Commit 信息 ...")
            //将消息信息，提交到本地消息池中！
            localMessagePool = append(localMessagePool, p.messagePool[c.Digest].Message)
            info := ""
            if p.node.nodeID != "N0" {
                info = p.node.nodeID + "节点已将 msgid:" + strconv.Itoa(p.messagePool[c.Digest].ID)
+ "存入本地消息池中,消息内容为：" + p.messagePool[c.Digest].Content
            } else {
                info = "主节点已将 msgid:" + strconv.Itoa(p.messagePool[c.Digest].ID) + "存入本地
消息池中,消息内容为：" + p.messagePool[c.Digest].Content
            }
            fmt.Println(info)
            fmt.Println("正在 reply 客户端 ...")
            tcpDial([]byte(info), p.messagePool[c.Digest].ClientAddr)
            p.isReply[c.Digest] = true
            fmt.Println("reply 完毕")
        }
        p.lock.Unlock()
    }
}
```

5. reply 阶段：节点接收到 2f+1 个 commit 信息（包含自己），并全部签名验证通过，则可以把消息存入到本地，并向客户端返回 reply 消息。

```
fmt.Println(info)
            fmt.Println("正在 reply 客户端 ...")
            tcpDial([]byte(info), p.messagePool[c.Digest].ClientAddr)
            p.isReply[c.Digest] = true
            fmt.Println("reply 完毕")
```

程序执行过程：

首先切换到项目根路径，分别运行下面两段代码进行初始化：

```
go mod init pbft

go build -o pbft.exe
```

开启五个端口（一个客户端，四个节点）

```
客户端执行 .\pbft.exe client

其他四个节点依次执行 .\pbft.exe N0  .\pbft.exe N1  .\pbft.exe N2  .\pbft.exe N3
```

客户端运行

```
PS D:\学习数据\Go项目\src\PBFT-Experiment-main\PBFT-Experiment-main> .\pbft.exe client
客户端开启监听，地址：127.0.0.1:8888
 ------------------------------------------------------------------------------------
|  已进入PBFT测试Demo客户端，请启动全部节点后再发送消息！ :)  |
 ------------------------------------------------------------------------------------
请在下方输入要存入节点的信息：
```

主节点运行

```
PS D:\学习数据\Go项目\src\PBFT-Experiment-main\PBFT-Experiment-main> .\pbft.exe N0
节点开启监听，地址：127.0.0.1:8000
```

三个从节点运行：

```
PS D:\学习数据\Go项目\src\PBFT-Experiment-main\PBFT-Experiment-main> .\pbft.exe N1
节点开启监听，地址：127.0.0.1:8001
```

1. 系统无拜占庭节点，均正常运行

1.1 request 阶段：

客户端运行：

```
请在下方输入要存入节点的信息：
x=1
{"Content":"x=1","ID":4927511167,"Timestamp":1684485861621541000,"ClientAddr":"127.0.0.1:8888"}
```

1.2 pre-prepare 阶段：

主节点运行：

```
主节点已接收到客户端发来的request ...
收到的request消息为:  {x=1 4927511167}
已将request存入临时消息池
Press enter to continue...
正在向其他节点进行进行PrePrepare广播 ...
PrePrepare消息内容为:  {{{x=1 4927511167} 1684485861621541000 127.0.0.1:8888} 7ca0fd68534765a666e3b86d5c7748abc21c2de336bdfea5b291e6c11bc
c08b6 1 [160 97 114 75 96 42 235 215 35 0 196 139 89 188 91 203 63 70 249 126 217 241 136 125 70 50 160 144 140 136 163 79 184 11 4 170 8
7 46 174 227 214 227 248 195 31 23 212 64 15 231 108 6 179 200 17 85 37 49 89 111 213 176 238 237 193 214 29 200 193 117 65 233 235 42 20
3 216 229 203 183 165 249 39 5 151 83 234 165 245 34 210 81 48 119 51 7 216 7 67 159 83 81 163 87 195 119 57 44 151 191 197 213 246 114 1
27 35 133 234 132 230 139 95 212 44 11 77 40 88 134]}
PrePrepare广播完成
Press enter to continue...
```

1.3 prepare 阶段

三个从节点运行

```
本节点已接收到主节点发来的PrePrepare ...
Press enter to continue...
已将消息存入临时节点池
正在进行Prepare广播 ...
广播的Prepare消息内容为:  {7ca0fd68534765a666e3b86d5c7748abc21c2de336bdfea5b291e6c11bcc08b6 1 N1 [37 228 50 170 178 205 70 241 230 71 5 1
5 245 50 83 56 28 151 223 83 189 228 1 206 204 45 12 11 103 107 137 70 195 202 49 18 13 12 219 216 168 251 163 159 93 140 32 5 211 122 14
8 136 198 72 235 146 221 144 28 247 80 36 80 230 66 163 76 246 148 158 92 206 81 41 72 36 151 224 147 138 134 5 108 117 215 129 83 214 18
5 165 135 195 49 183 174 139 14 116 38 202 172 239 83 141 62 62 237 236 59 89 32 79 42 48 240 236 35 190 235 177 17 136 244 143 194 21 88
 73]}
Prepare广播完成
Press enter to continue...
```

主节点运行

```
本节点已接收到N1节点发来的Prepare ...
Press enter to continue...
本节点已接收到N2节点发来的Prepare ...
Press enter to continue...
本节点已收到至少2f个节点(包括本地节点)发来的Prepare信息，内容为:  &{7ca0fd68534765a666e3b86d5c7748abc21c2de336bdfea5b291e6c11bcc08b6 1 N
2 [113 94 226 95 135 3 67 77 45 82 215 132 150 61 65 42 210 128 242 165 71 116 211 193 31 145 20 254 167 143 10 83 81 145 199 28 210 142
240 54 16 166 144 253 117 75 1 175 29 168 118 53 85 90 182 116 158 66 249 164 254 141 72 111 91 66 129 187 227 156 86 54 127 76 233 144 2
16 253 79 71 242 8 24 118 119 190 60 225 187 49 229 24 212 222 237 229 241 66 156 33 122 65 21 249 222 24 7 21 192 48 114 218 21 162 206
100 92 3 200 169 70 88 70 248 232 65 53 220]}
```

1.4 commit 阶段

主节点运行：

```
正在进行commit广播 ...
广播的commit消息内容为:  [123 34 68 105 103 101 115 116 34 58 34 55 99 97 48 102 100 54 56 53 51 52 55 54 53 97 54 54 54 101 51 98 56 54
100 53 99 55 55 52 56 97 98 99 50 49 99 50 100 101 51 51 54 98 100 102 101 97 53 98 50 57 49 101 54 99 49 49 98 99 99 48 56 98 54 34 44 3
4 83 101 113 117 101 110 99 101 73 68 34 58 49 44 34 78 111 100 101 73 68 34 58 34 78 48 34 44 34 83 105 103 110 34 58 34 111 71 70 121 8
3 50 65 113 54 57 99 106 65 77 83 76 87 98 120 98 121 122 57 71 43 88 55 90 56 89 104 57 82 106 75 103 107 73 121 73 111 48 43 52 67 119
83 113 86 121 54 117 52 57 98 106 43 77 77 102 70 57 82 65 68 43 100 115 66 114 80 73 69 86 85 108 77 86 108 118 49 98 68 117 55 99 72 87
 72 99 106 66 100 85 72 112 54 121 114 76 50 79 88 76 116 54 88 53 74 119 87 88 85 43 113 108 57 83 76 83 85 84 66 51 77 119 102 89 66 48
 79 102 85 49 71 106 86 56 78 51 79 83 121 88 118 56 88 86 57 110 74 47 73 52 88 113 104 79 97 76 88 57 81 115 67 48 48 111 87 73 89 61 3
4 125]
commit广播完成
```

三个从节点运行：

```
本节点已接收到N2节点发来的Prepare ...
Press enter to continue...
本节点已收到至少2f个节点(包括本地节点)发来的Prepare信息，内容为:  &{7ca0fd68534765a666e3b86d5c7748abc21c2de336bdfea5b291e6c11bcc08b6 1 N
2 [113 94 226 95 135 3 67 77 45 82 215 132 150 61 65 42 210 128 242 165 71 116 211 193 31 145 20 254 167 143 10 83 81 145 199 28 210 142
240 54 16 166 144 253 117 75 1 175 29 168 118 53 85 90 182 116 158 66 249 164 254 141 72 111 91 66 129 187 227 156 86 54 127 76 233 144 2
16 253 79 71 242 8 24 118 119 190 60 225 187 49 229 24 212 222 237 229 241 66 156 33 122 65 21 249 222 24 7 21 192 48 114 218 21 162 206
100 92 3 200 169 70 88 70 248 232 65 53 220]}
正在进行commit广播 ...
广播的commit消息内容为:  [123 34 68 105 103 101 115 116 34 58 34 55 99 97 48 102 100 54 56 53 51 52 55 54 53 97 54 54 54 101 51 98 56 54
100 53 99 55 55 52 56 97 98 99 50 49 99 50 100 101 51 51 54 98 100 102 101 97 53 98 50 57 49 101 54 99 49 49 98 99 99 48 56 98 54 34 44 3
4 83 101 113 117 101 110 99 101 73 68 34 58 49 44 34 78 111 100 101 73 68 34 58 34 78 49 34 44 34 83 105 103 110 34 58 34 74 101 81 121 1
13 114 76 78 82 118 72 109 82 119 85 80 57 84 74 84 79 66 121 88 51 49 79 57 53 65 72 79 122 67 48 77 67 50 100 114 105 85 98 68 121 106
69 83 68 81 122 98 50 75 106 55 111 53 57 100 106 67 65 70 48 51 113 85 105 77 90 73 54 53 76 100 107 66 122 51 85 67 82 81 53 107 75 106
 84 80 97 85 110 108 122 79 85 83 108 73 74 74 102 103 107 52 113 71 66 87 120 49 49 52 70 84 49 114 109 108 104 56 77 120 116 54 54 76 6
8 110 81 109 121 113 122 118 85 52 48 43 80 117 51 115 79 49 107 103 84 121 111 119 56 79 119 106 118 117 117 120 69 89 106 48 106 56 73
86 87 69 107 61 34 125]
commit广播完成
```

1.5 reply 阶段

主节点运行：

```
Press enter to continue...
本节点已接收到N3节点发来的Prepare ...
Press enter to continue...
本节点已接收到N1节点发来的Commit ...
本节点已接收到N2节点发来的Commit ...
本节点已收到至少2f + 1 个节点(包括本地节点)发来的Commit信息 ...
主节点已将msgid:4927511167存入本地消息池中,消息内容为：x=1
正在reply客户端 ...
reply完毕
本节点已接收到N3节点发来的Commit ...
```

三个从节点运行：

```
Press enter to continue...
本节点已接收到N3节点发来的Prepare ...
Press enter to continue...
本节点已接收到N0节点发来的Commit ...
本节点已接收到N2节点发来的Commit ...
本节点已收到至少2f + 1 个节点(包括本地节点)发来的Commit信息 ...
N1节点已将msgid:4927511167存入本地消息池中,消息内容为：x=1
正在reply客户端 ...
reply完毕
本节点已接收到N3节点发来的Commit ...
```

客户端收到消息：

```
{"Content":"x=1","ID":4927511167,"Timestamp":1684485861621541000,"ClientAddr":"127.0.0.1:8888"}
主节点已将msgid:4927511167存入本地消息池中,消息内容为：x=1
N1节点已将msgid:4927511167存入本地消息池中,消息内容为：x=1
N2节点已将msgid:4927511167存入本地消息池中,消息内容为：x=1
N3节点已将msgid:4927511167存入本地消息池中,消息内容为：x=1
```

2. 系统存在一个拜占庭节点，其余两个从节点均正常运行

2.1 request 阶段：

客户端运行：

```
请在下方输入要存入节点的信息：
x=2
{"Content":"x=2","ID":1601701342,"Timestamp":1684486948102588900,"ClientAddr":"127.0.0.1:8888"}
```

2.2 pre-prepare 阶段：

主节点运行：

```
Press enter to continue...
正在向其他节点进行进行PrePrepare广播 ...
PrePrepare消息内容为: {{{x=2 1601701342} 1684486948102588900 127.0.0.1:8888} d01d58ecad78ec8f20a1bf61dc963dadc83cf94d7d297d9707f225351df
db4fb 1 [50 1 140 181 131 96 164 69 247 161 93 184 198 113 56 203 168 25 162 43 84 13 44 70 16 166 116 113 199 176 62 145 98 90 9 211 148
 100 145 93 246 200 67 162 163 29 80 180 244 245 131 173 190 93 12 211 3 211 146 62 87 117 205 55 43 174 39 100 134 104 60 218 109 136 14
4 112 147 213 187 96 117 88 238 68 110 42 8 56 106 239 210 40 117 172 30 196 35 230 108 83 115 50 141 140 92 190 103 13 196 185 156 200 1
2 192 224 255 68 160 226 242 5 141 83 198 128 240 178 17]}
PrePrepare广播完成
```

2.3 prepare 阶段

三个从节点运行

```
本节点已接收到主节点发来的PrePrepare ...
Press enter to continue...
已将消息存入临时节点池
正在进行Prepare广播 ...
广播的Prepare消息内容为: {d01d58ecad78ec8f20a1bf61dc963dadc83cf94d7d297d9707f225351dfdb4fb 1 N2 [37 211 137 251 245 106 174 168 24 47 11
5 94 156 223 75 208 159 87 95 137 157 2 148 209 140 87 170 102 143 164 243 198 239 81 123 31 7 201 40 243 174 133 67 97 46 81 73 125 201
9 101 213 175 141 211 249 127 219 169 222 143 191 25 187 208 204 24 226 36 17 233 16 56 235 95 223 197 99 43 148 38 227 3 48 209 195 144
58 21 39 177 227 95 36 48 229 146 186 29 9 245 171 189 154 55 185 135 215 135 207 210 101 212 171 134 44 143 208 102 182 65 81 55 33 141
46 157 29]}
Prepare广播完成
```

主节点运行

```
本节点已接收到N1节点发来的Prepare ...
Press enter to continue...
本节点已接收到N2节点发来的Prepare ...
Press enter to continue...
本节点已收到至少2f个节点(包括本地节点)发来的Prepare信息，内容为:  &{d01d58ecad78ec8f20a1bf61dc963dadc83cf94d7d297d9707f225351dfdb4fb 1 N
2 [37 211 137 251 245 106 174 168 24 47 115 94 156 223 75 208 159 87 95 137 157 2 148 209 140 87 170 102 143 164 243 198 239 81 123 31 7
201 40 243 174 133 67 97 46 81 73 125 201 9 101 213 175 141 211 249 127 219 169 222 143 191 25 187 208 204 24 226 36 17 233 16 56 235 95
223 197 99 43 148 38 227 3 48 209 195 144 58 21 39 177 227 95 36 48 229 146 186 29 9 245 171 189 154 55 185 135 215 135 207 210 101 212 1
71 134 44 143 208 102 182 65 81 55 33 141 46 157 29]}
```

2.4 commit 阶段

主节点运行：

```
正在进行commit广播 ...
广播的commit消息内容为: [123 34 68 105 103 101 115 116 34 58 34 100 48 49 100 53 56 101 99 97 100 55 56 101 99 56 102 50 48 97 49 98 102
 54 49 100 99 57 54 51 100 97 100 99 56 51 99 102 57 52 100 55 100 50 57 55 100 57 55 48 55 102 50 50 53 51 53 49 100 102 100 98 52 102 9
8 34 44 34 83 101 113 117 101 110 99 101 73 68 34 58 49 44 34 78 111 100 101 73 68 34 58 34 78 48 34 44 34 83 105 103 110 34 58 34 77 103
 71 77 116 89 78 103 112 69 88 51 111 86 50 52 120 110 69 52 121 54 103 90 111 105 116 85 68 83 120 71 69 75 90 48 99 99 101 119 80 112 7
0 105 87 103 110 84 108 71 83 82 88 102 98 73 81 54 75 106 72 86 67 48 57 80 87 68 114 98 53 100 68 78 77 68 48 53 73 43 86 51 88 78 78 1
21 117 117 74 50 83 71 97 68 122 97 98 89 105 81 99 74 80 86 117 50 66 49 87 79 53 69 98 105 111 73 79 71 114 118 48 105 104 49 114 66 55
 69 73 43 90 115 85 51 77 121 106 89 120 99 118 109 99 78 120 76 109 99 121 65 122 65 52 80 57 69 111 79 76 121 66 89 49 84 120 111 68 11
9 115 104 69 61 34 125]
commit广播完成
```

三个从节点运行：

```
本节点已接收到N1节点发来的Prepare ...
Press enter to continue...
本节点已收到至少2f个节点(包括本地节点)发来的Prepare信息，内容为:  &{d01d58ecad78ec8f20a1bf61dc963dadc83cf94d7d297d9707f225351dfdb4fb 1 N
1 [115 55 140 53 160 247 76 135 225 168 28 187 226 114 62 73 75 78 53 20 96 179 92 38 124 215 28 19 4 176 171 53 30 255 161 69 29 189 34
206 151 12 119 20 205 206 159 33 228 13 183 22 156 89 76 43 114 246 140 101 39 31 160 43 164 198 148 151 56 96 243 166 188 176 178 127 18
0 134 116 210 206 217 93 56 228 217 63 39 83 171 214 111 50 71 47 95 183 40 145 92 6 28 63 18 26 58 99 123 220 68 249 40 95 55 166 180 65
 206 24 45 228 88 172 156 153 116 82 240]}
正在进行commit广播 ...
广播的commit消息内容为: [123 34 68 105 103 101 115 116 34 58 34 100 48 49 100 53 56 101 99 97 100 55 56 101 99 56 102 50 48 97 49 98 102
 54 49 100 99 57 54 51 100 97 100 99 56 51 99 102 57 52 100 55 100 50 57 55 100 57 55 48 55 102 50 50 53 51 53 49 100 102 100 98 52 102 9
8 34 44 34 83 101 113 117 101 110 99 101 73 68 34 58 49 44 34 78 111 100 101 73 68 34 58 34 78 50 34 44 34 83 105 103 110 34 58 34 74 100
 79 74 43 47 86 113 114 113 103 89 76 51 78 101 110 78 57 76 48 74 57 88 88 52 109 100 65 112 84 82 106 70 101 113 90 111 43 107 56 56 98
 118 85 88 115 102 66 56 107 111 56 54 54 70 81 50 69 117 85 85 108 57 121 81 108 108 49 97 43 78 48 47 108 47 50 54 110 101 106 55 56 90
 117 57 68 77 71 79 73 107 69 101 107 81 79 79 116 102 51 56 86 106 75 53 81 109 52 119 77 119 48 99 79 81 79 104 85 110 115 101 78 102 7
4 68 68 108 107 114 111 100 67 102 87 114 118 90 111 51 117 89 102 88 104 56 47 83 90 100 83 114 104 105 121 80 48 71 97 50 81 86 69 51 7
3 89 48 117 110 82 48 61 34 125]
commit广播完成
```

2.5 reply 阶段

主节点运行：

```
本节点已接收到N1节点发来的Commit ...
本节点已接收到N2节点发来的Commit ...
本节点已收到至少2f + 1 个节点(包括本地节点)发来的Commit信息 ...
主节点已将msgid:1601701342存入本地消息池中,消息内容为: x=2
正在reply客户端 ...
reply完毕
```

三个从节点运行：

```
本节点已接收到N0节点发来的Commit ...
本节点已接收到N2节点发来的Commit ...
本节点已收到至少2f + 1 个节点(包括本地节点)发来的Commit信息 ...
N1节点已将msgid:1601701342存入本地消息池中,消息内容为: x=2
正在reply客户端 ...
reply完毕
```

客户端收到消息：

```
请在下方输入要存入节点的信息:
x=2
{"Content":"x=2","ID":1601701342,"Timestamp":1684486948102588900,"ClientAddr":"127.0.0.1:8888"}
主节点已将msgid:1601701342存入本地消息池中,消息内容为: x=2
N1节点已将msgid:1601701342存入本地消息池中,消息内容为: x=2
N2节点已将msgid:1601701342存入本地消息池中,消息内容为: x=2
```

可以看到，客户端依然会接收到 reply，因为根据公式 n >= 3f+1 ，就算宕机一个节点，系统依然能顺利运行

3. 系统存在两个拜占庭节点，剩下一个从节点正常运行

3.1 request 阶段：

客户端运行：

```
PS D:\学习数据\Go项目\src\PBFT-Experiment-main\PBFT-Experiment-main> .\pbft.exe client
客户端开启监听，地址：127.0.0.1:8888
 ----------------------------------------------------------------------------
|  已进入PBFT测试Demo客户端，请启动全部节点后再发送消息！ :)  |
 ----------------------------------------------------------------------------
请在下方输入要存入节点的信息:
x=3
{"Content":"x=3","ID":5146891951,"Timestamp":1684487256518545500,"ClientAddr":"127.0.0.1:8888"}
```

3.2 pre-prepare 阶段：

主节点运行：

```
主节点已接收到客户端发来的request ...
收到的request消息为: {x=3 5146891951}
已将request存入临时消息池
Press enter to continue...
正在向其他节点进行进行PrePrepare广播 ...
PrePrepare消息内容为: {{{x=3 5146891951} 1684487256518545500 127.0.0.1:8888} 0e310e4858040e9cede12eaedc424681c162e0edca534a4aa9fb874aca6
aec34 1 [174 149 69 59 86 31 113 10 87 116 216 52 175 33 178 247 50 217 38 241 206 70 15 122 12 78 106 71 231 114 157 28 7 65 51 201 107
179 218 40 2 121 206 198 96 93 167 111 200 244 48 137 77 240 161 33 145 41 193 211 232 172 55 220 32 131 25 153 86 178 85 53 238 249 180
122 161 209 185 63 211 131 211 220 150 189 226 193 12 44 195 129 143 3 110 196 139 7 253 96 59 116 55 76 113 229 114 76 136 55 247 107 22
4 157 56 161 98 27 97 149 175 145 35 107 142 8 240 42]}
PrePrepare广播完成
```

3.3 prepare 阶段

三个从节点运行

```
本节点已接收到主节点发来的PrePrepare ...
Press enter to continue...
已将消息存入临时节点池
正在进行Prepare广播 ...
广播的Prepare消息内容为: {0e310e4858040e9cede12eaedc424681c162e0edca534a4aa9fb874aca6aec34 1 N1 [149 163 120 16 194 223 146 65 97 147 16
2 141 165 32 152 130 243 26 59 205 95 241 161 136 40 188 156 169 203 115 245 112 158 30 161 105 123 37 44 47 226 3 170 247 241 30 39 217
81 40 20 159 193 84 83 227 132 55 42 0 112 170 54 60 186 151 84 226 200 3 111 184 58 60 220 128 121 190 57 218 230 46 40 103 5 76 6 213 6
3 149 45 172 183 130 238 248 48 211 107 191 131 213 154 22 116 243 44 221 17 23 51 194 130 158 175 253 224 159 108 45 56 223 41 92 58 116
 177 93]}
Prepare广播完成
```

主节点运行

```
本节点已接收到N1节点发来的Prepare ...
Press enter to continue...
```

可以看到，关闭两个节点后，故障节点已经超出了 pbft 的允许数量，消息进行到 Prepare 阶段由于接收不到满足数量的信息，固系统不再进行 commit 确认，客户端也接收不到 reply