Practical Byzantine Fault Tolerance

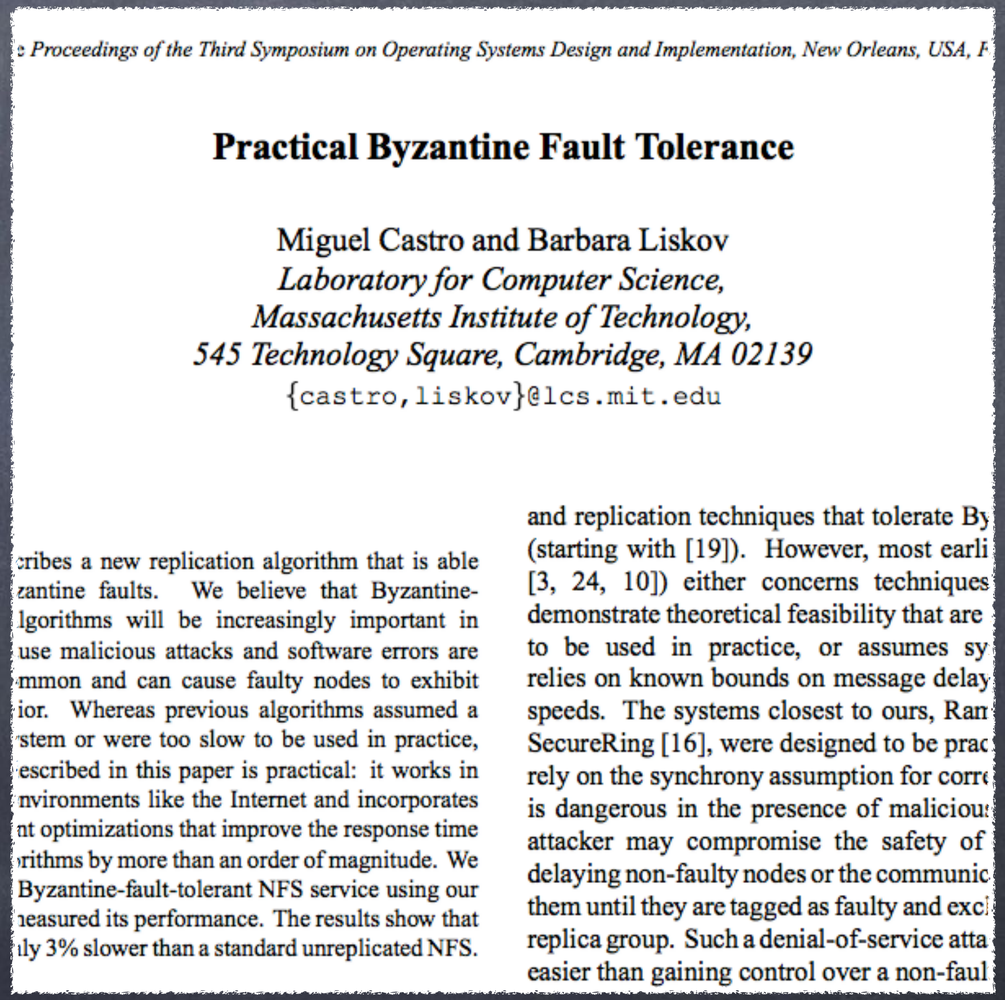
— the consensus algorithm in fabric

一致性问题引发的讨论

- 一致性是指对于分布式系统中的多个服务节点，给定一系列操作，在协议保障下，试图使得它们对处理结果达成某种程度的一致
- 要保障系统满足不同程度的一致性，往往需要通过共识算法来达成
- 拜占庭问题：Leslie Lamport 1982 年提出用来解释一致性问题的一个虚构模型

Practical Byzantine Fault Tolerance

- PBFT - 实用拜占庭容错算法
- Miguel Castro、Barbara Liskov 在 1999 年提出
- 解决的问题：提高了 BFT 效率，使得在实际应用中变得可行
- 目前 fabric 最为推荐的共识算法



e Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, F

Practical Byzantine Fault Tolerance

Miguel Castro and Barbara Liskov
Laboratory for Computer Science,
Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139
{castro,liskov}@lcs.mit.edu

cribes a new replication algorithm that is able
zantine faults. We believe that Byzantine-
lgorithms will be increasingly important in
use malicious attacks and software errors are
mmon and can cause faulty nodes to exhibit
ior. Whereas previous algorithms assumed a
stem or were too slow to be used in practice,
escribed in this paper is practical: it works in
nvironments like the Internet and incorporates
nt optimizations that improve the response time
rithms by more than an order of magnitude. We
Byzantine-fault-tolerant NFS service using our
neasured its performance. The results show that
ıly 3% slower than a standard unreplicated NFS.

and replication techniques that tolerate By
(starting with [19]). However, most earli
[3, 24, 10]) either concerns techniques
demonstrate theoretical feasibility that are
to be used in practice, or assumes sy
relies on known bounds on message delay
speeds. The systems closest to ours, Ran
SecureRing [16], were designed to be prac
rely on the synchrony assumption for corre
is dangerous in the presence of malicious
attacker may compromise the safety of
delaying non-faulty nodes or the communic
them until they are tagged as faulty and excl
replica group. Such a denial-of-service atta
easier than gaining control over a non-faul

只要系统中有 2/3 的节点是正常工作的，则可以保证一致性。

PBFT 算法描述

- View
- Replica
  - Primary : $p = v \bmod |R|$
  - Backups

PBFT 算法描述

- 1. 客户端向主节点发送请求调用服务操作
- 2. 主节点通过广播将请求发送给其他副本
- 3. 所有副本都执行请求并将结果发回客户端
- 4. 客户端需要等待 $f+1$ 个不同副本节点发回相同的结果，作为整个操作的最终结果

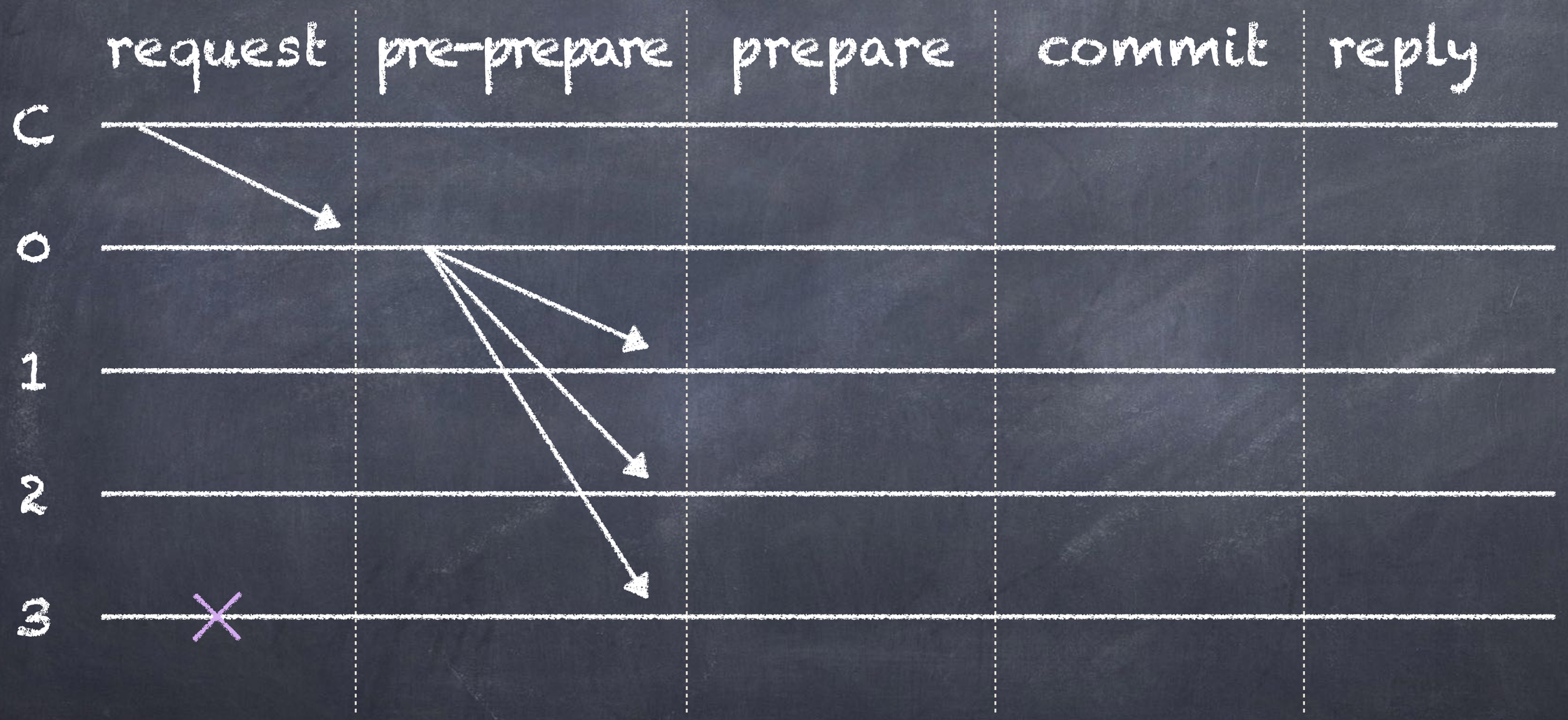
request
pre-prepare
prepare
commit
reply
C
0
1
2
3

PBFT 算法主线流程

- 主节点 p 收到客户端的请求 m 后，通过三阶段达成共识：
  - 预准备 pre-prepare
  - 准备 prepare
  - 确认 commit

完成请求排序

保证排序不变


request
pre-prepare
prepare
commit
reply
C
0
1
2
3

pre-prepare

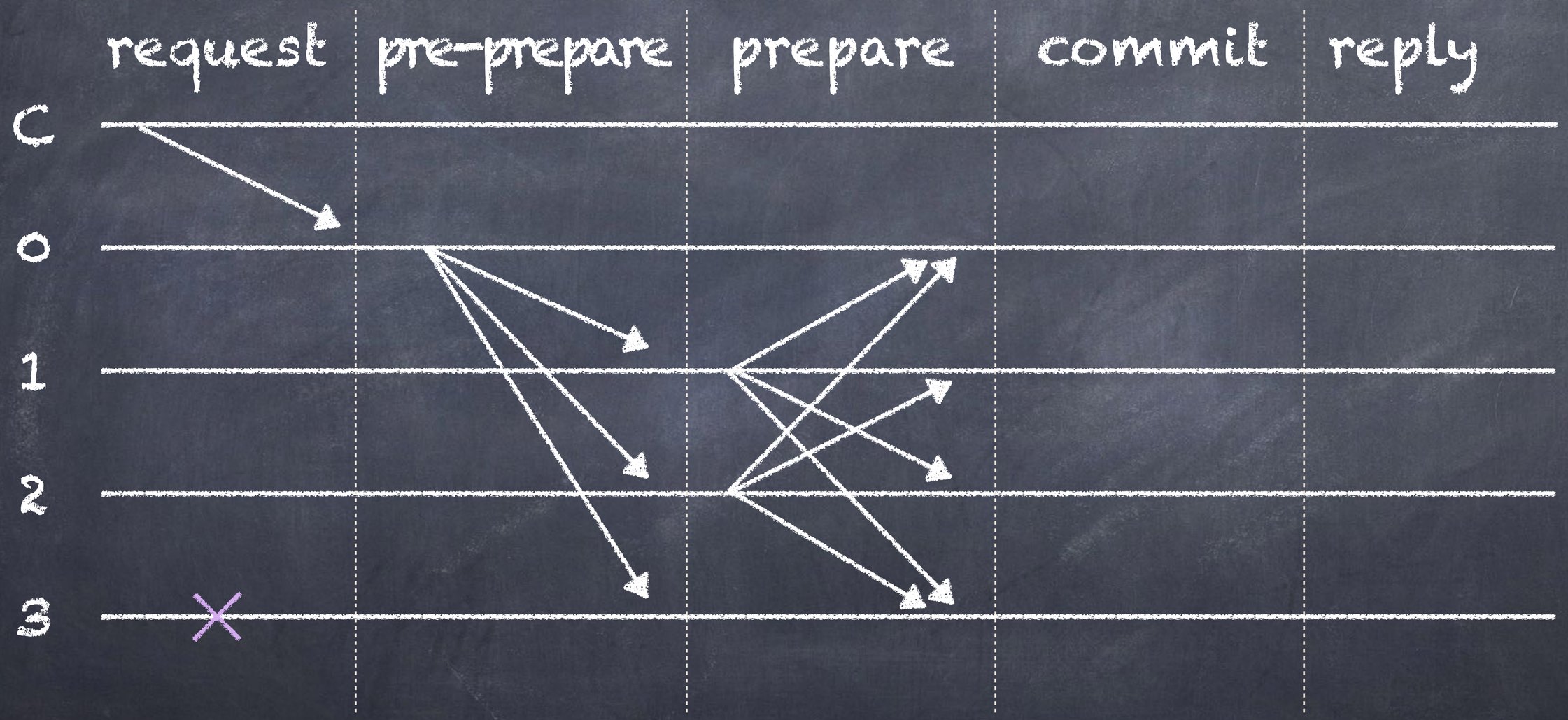
- 主节点为请求分配序号 n，向所有备份节点广播预准备消息：
  - <<PRE-PREPARE, v, n, d>, m>
    - v：视图编号
    - m：客户端发送到请求消息
    - d：请求消息 m 的摘要

pre-prepare

- 备份节点 i 收到预准备消息，满足以下条件才会接受：
  - 1. 请求和预准备消息的签名正确，并且 d 与 m 的摘要一致
  - 2. 当前视图编号是 v
  - 3. 该备份节点从未在视图 v 中接受过序号为 n 但是摘要 d 不同的消息 m
  - 4. 预准备消息的序号 n 必须在水线的上下限 h 和 H 之间

pre-prepare

- 备份节点 i 接受预准备消息后，该节点进入准备阶段，并向所有副本节点发送准备消息：
  - <PREPARE, v, n, d, i>
  - 同时，将预准备消息和准备消息写入自己的消息日志
- 如果预准备消息验证未通过，节点则什么也不做

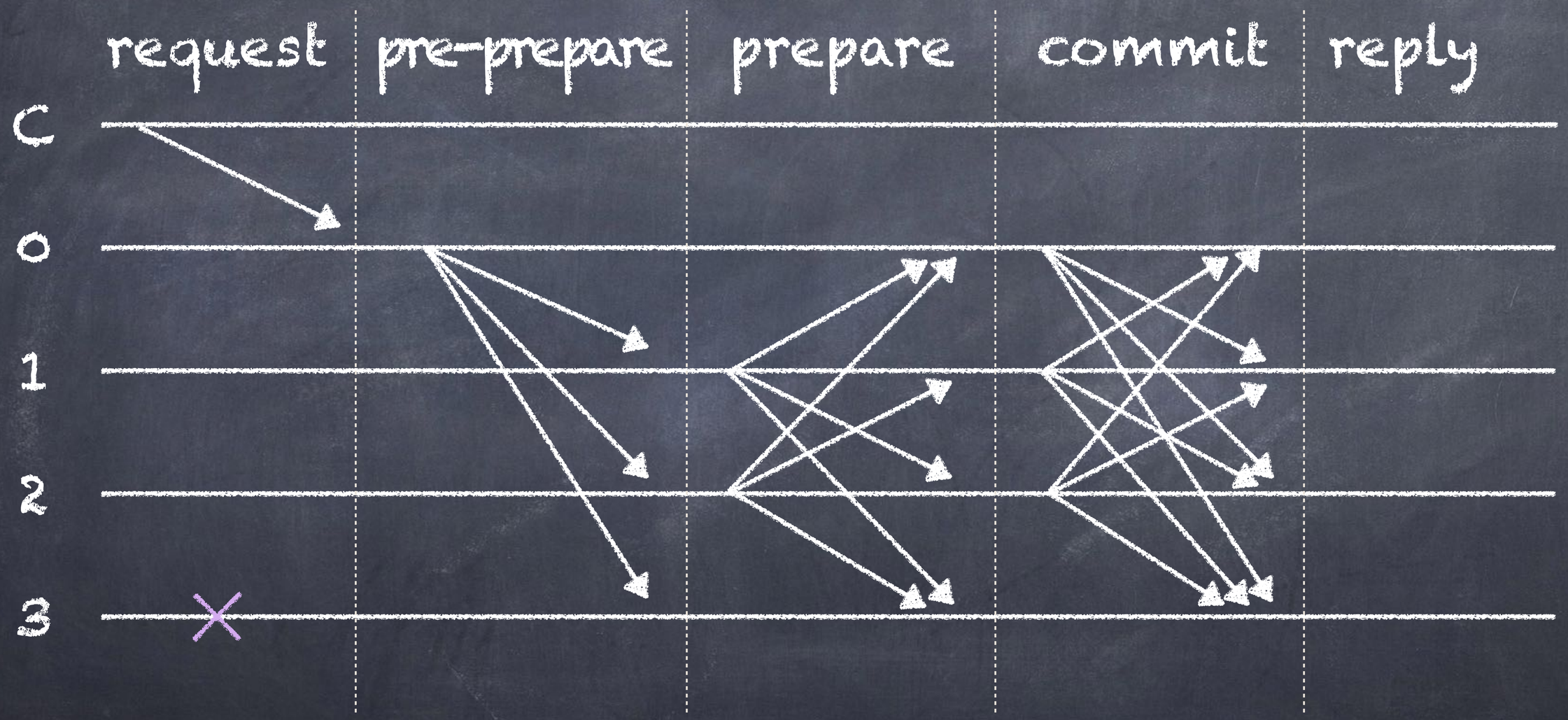

request
pre-prepare
prepare
commit
reply
C
0
1
2
3

prepare

- 所有副本节点在收到准备消息后，验证：
  - 消息签名
  - 视图编号
  - 消息序号
- 验证通过后，将准备消息写入消息日志

prepare

- 副本节点 i 准备阶段完成的标志：
  - 副本节点 i 将 (m, v, n, i) 记入其消息日志中
  - 从 2f 个不同副本节点收到的与预准备消息一致的准备消息
- 预准备阶段和准备阶段确保所有正常节点对同一个视图中的请求序号达成一致

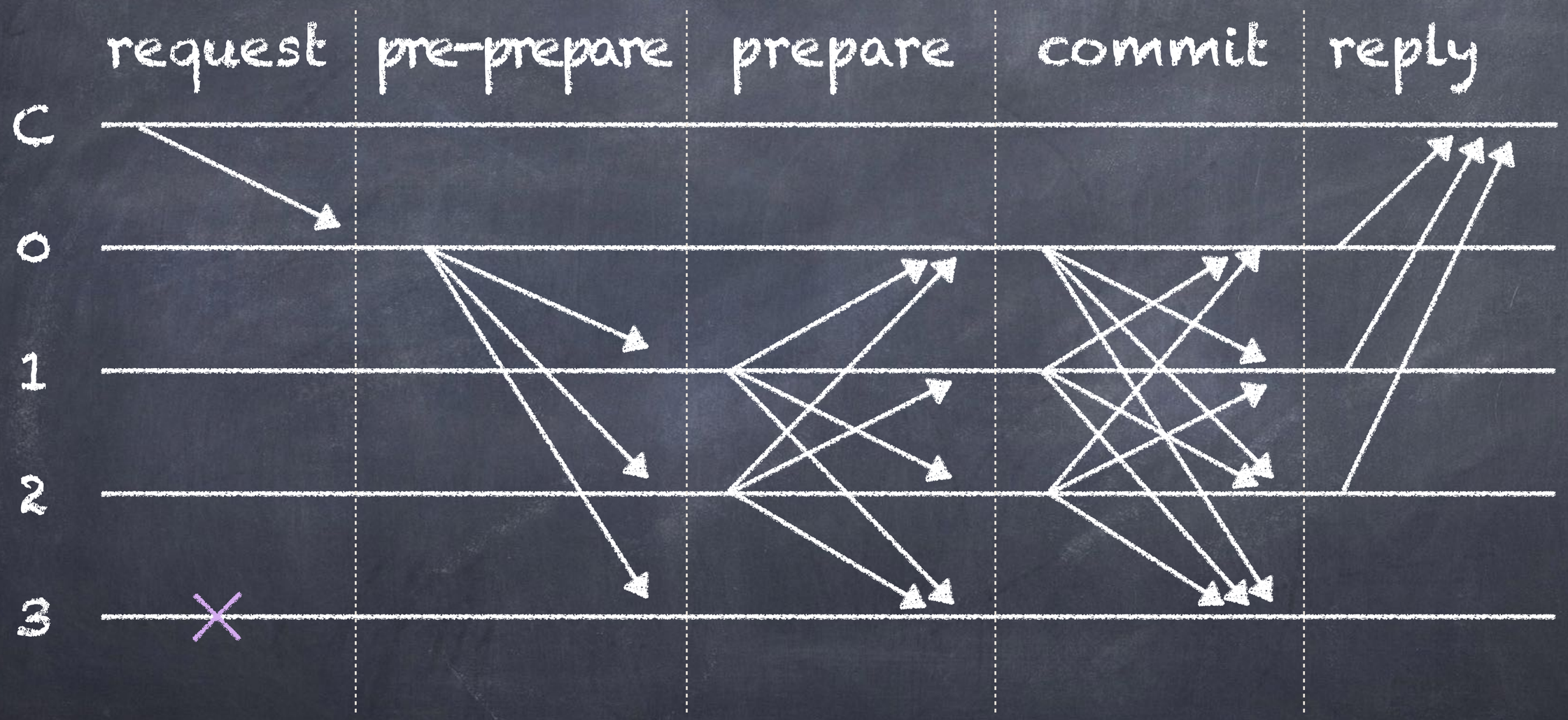

request
pre-prepare
prepare
commit
reply
C
0
1
2
3

commit

- 副本节点 i 进入确认阶段，向其他副本节点广播确认消息：
  - <COMMIT, v, n, D(m), i>
- 其他副本节点验证：
  - 消息签名
  - 视图编号
  - 消息序号

commit

- 副本节点 i 接受确认消息的条件：
  - prepared(m, v, n, i) 为真，并且 i 接收到另外的 2f 个确认
- 确认完成的标志：
  - 任意 f + 1 个正常副本节点集合中的所有副本 i 其 prepared(m, v, n, i) 为真


request
pre-prepare
prepare
commit
reply
C
0
1
2
3

最终……

- 副本节点 i 接受确认消息后，执行消息 m 中的请求，向客户的发送回复

如果主节点失效，怎么办？

- 使用计时器的超时机制触发视图变更事件

"Thank you"

- Gong Yaming <ymgongcn#gmail.com>