Fehlertolerante, konsistente Replikation mit Hilfe des PAXOS-Algorithmus

Zweck des Algorithmus

- System von Prozessen, die Werte vorschlagen können
- Einigung auf genau einen der Vorgeschlagenen Werte
- Etwas formaler ausgedrückt: System von N Prozessen mit je einem Wert input[i]. Ziel des Algorithmus: Die Prozesse einigen sich auf einen Wert, so dass alle als output[i] entweder den gleichen Wert aus input[] haben, oder “NULL”.
- Beliebige Zahl von nicht-byzantinischen Fehlern kann toleriert werden

Geschichte

- entwickelt von L. Lamport 1989
- Auslöser: ECHO Dateisystem
- Ursprünglich als Erfindung die Anfang des letzten Jahrtausends auf der Insel Paxos gemacht wurde ausgegeben:
  - Part-time parliament
  - ungewöhnlich ehrliche Politiker
  - Alles wo er sich nicht festlegen wollte, “hat man bei den Ausgrabungen noch nicht entdeckt”
- Fast keiner verstand seine Beschreibung
- Im Januar 1990 an ein Computermagazin zur Veröffentlichung gesandt, aber erst 1998 dort veröffentlicht

Voraussetzungen / was darf nicht passieren

- **KEINE BYZANTINISCHEN FEHLER**
  eine einzige korrupte Nachricht kann das System zum Zusammenbruch bringen
- Fehlverhalten: Fail Stop
- Wiederanlauf erlaubt
- Direkte Kommunikation möglich
- Alle beteiligten Prozesse müssen bekannt sein
- Mehrheit muss festgelegt sein
- nichtflüchtiger Speicher
- irgendeine Form von Timer

Voraussetzungen / was darf passieren

- Prozesse dürfen in beliebiger Anzahl und Häufigkeit ausfallen
- Nachrichten dürfen in beliebiger Anzahl verloren gehen, verzögert werden und sogar vertauscht werden.
- Die Prozesse dürfen mit beliebiger Geschwindigkeit operieren

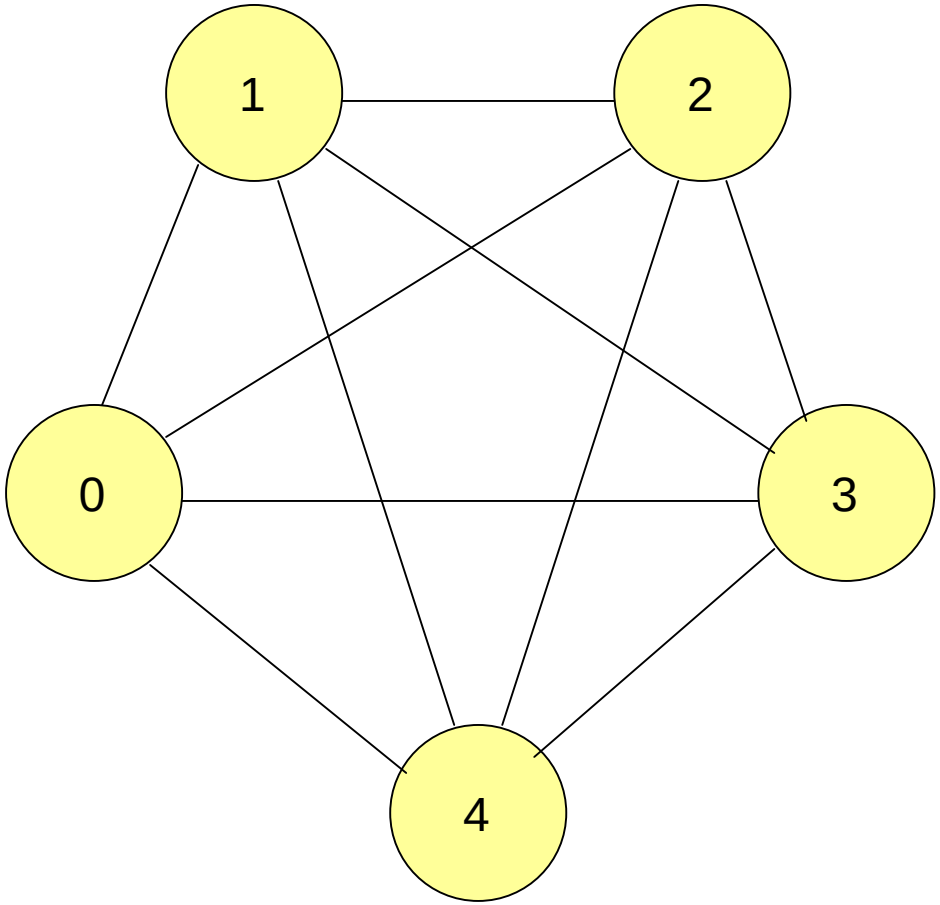
Voraussetzungen / was sollte passieren

Die Konsistenz bleibt zwar auch erhalten, wenn die Prozesse gar nichts tun, aber das Ziel des Algorithmus ist es ja eigentlich eine Einigung zu erzielen.

- Nachrichten sollten so schnell wie möglich bearbeitet / beantwortet werden
- Fortschritt ist nur dann möglich, wenn eine Mehrzahl der Prozesse miteinander kommunizieren kann.
- Fortschritt ist nur dann sichergestellt, wenn es genau einen Anführer gibt.

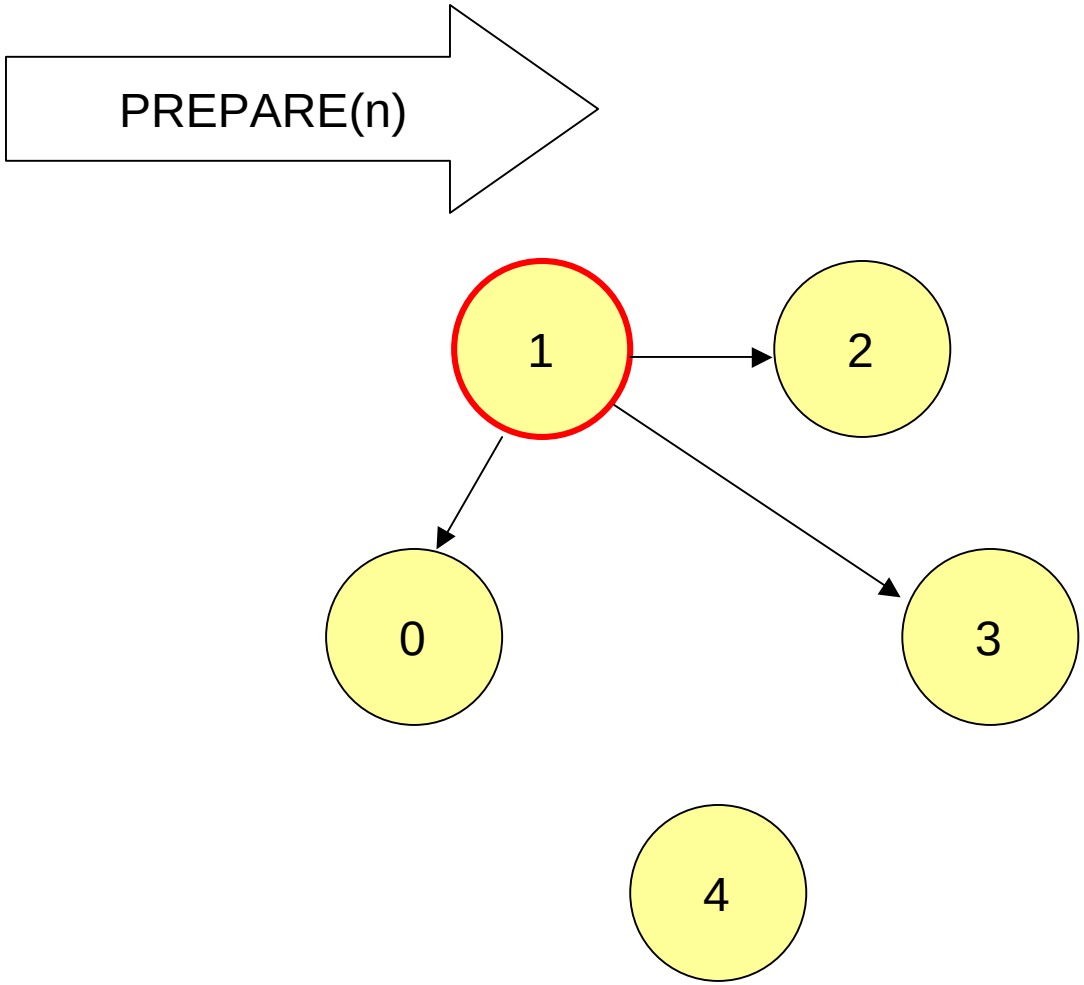
(Basic-)Paxos (1)

- Jeder Prozess kann Proposer, Acceptor und Learner sein



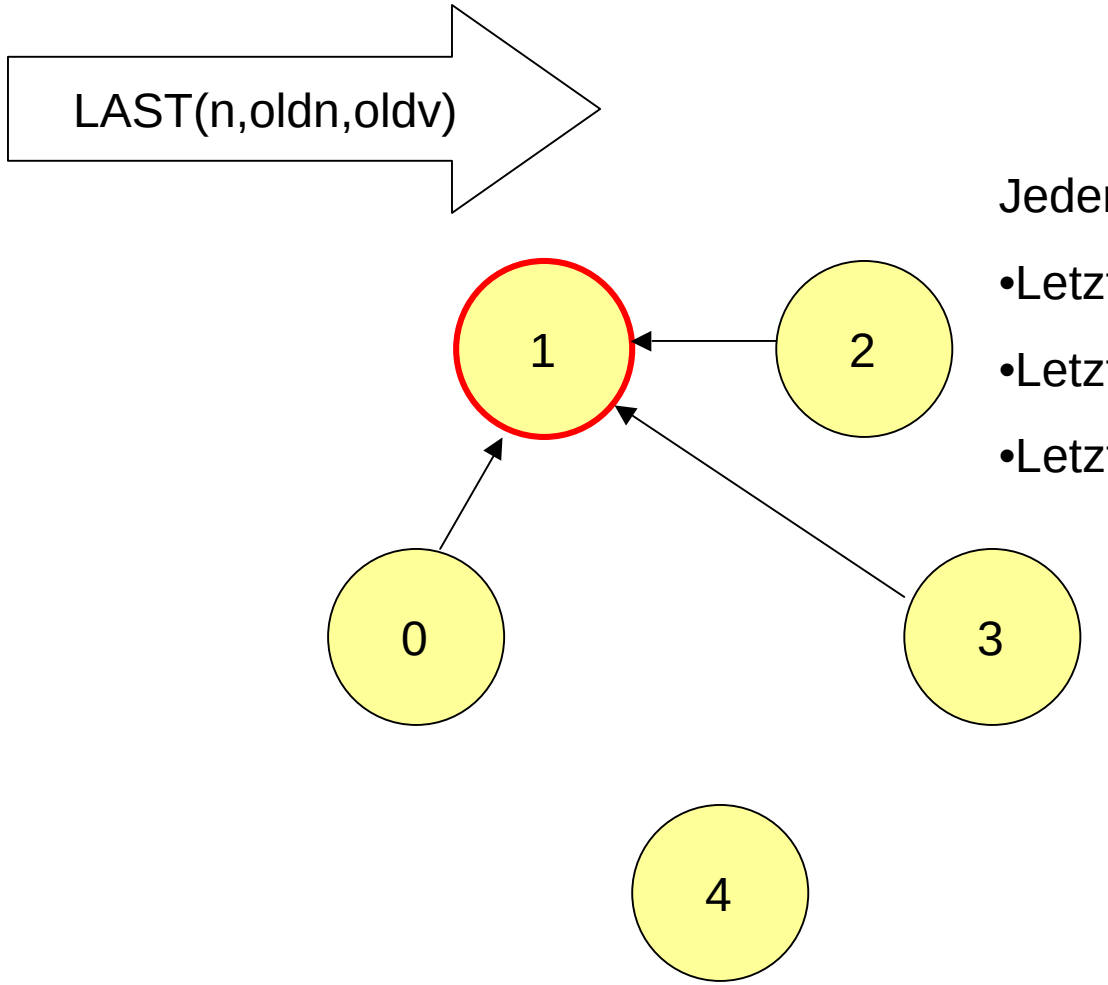
Paxos (2)

1. Phase: “PREPARE”



Paxos (3)

Antwort auf PREPARE, falls n>lastprepare

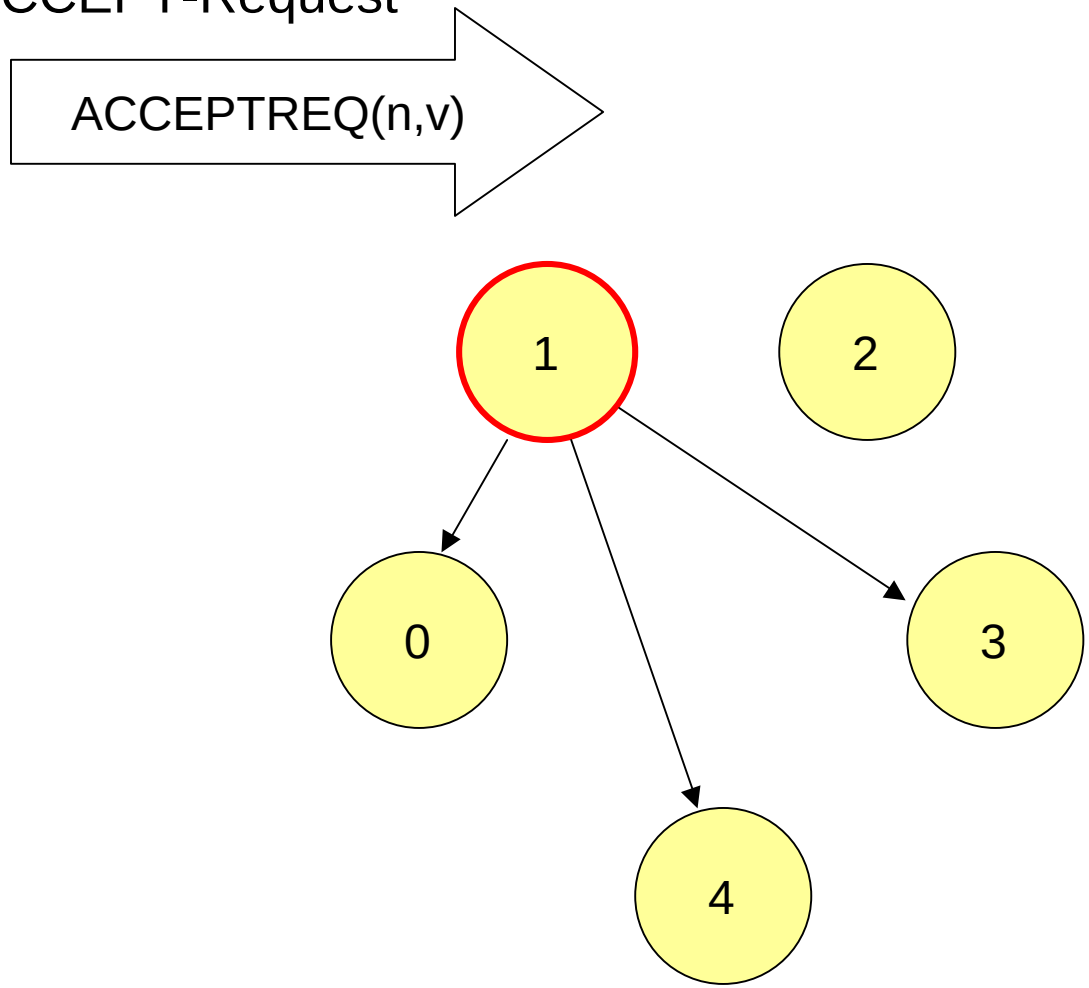


Jeder Prozess merkt sich:

- Letzte PREPARE-Runde
- Letzte akzeptierte Runde
- Letzter akzeptierter Wert

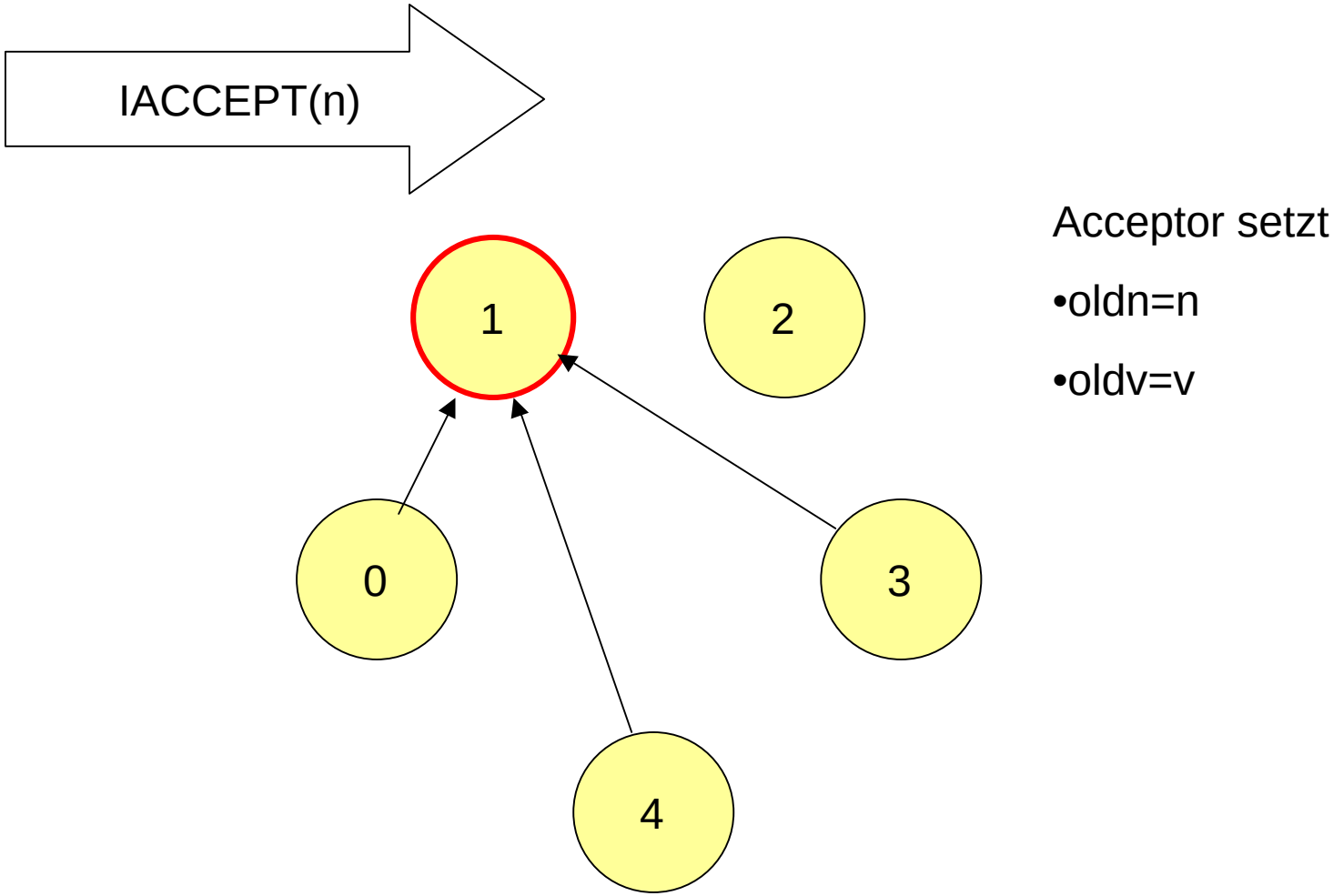
Paxos (4)

2. Phase: Falls PREPARE von Mehrheit akzeptiert: ACCEPT-Request



Paxos (5)

Antwort auf ACCEPTREQ, falls n>=lastPrepare

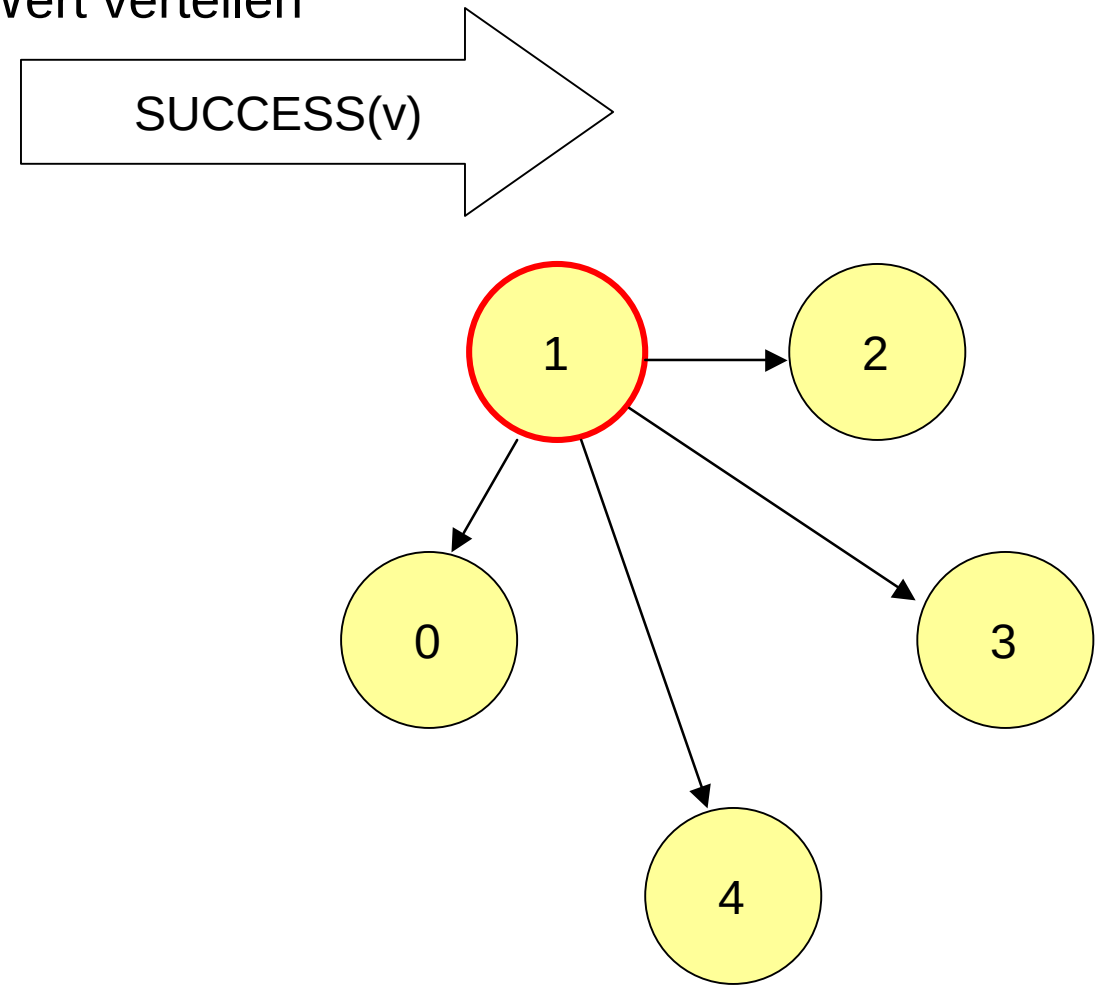


Acceptor setzt

- oldn=n
- oldv=v

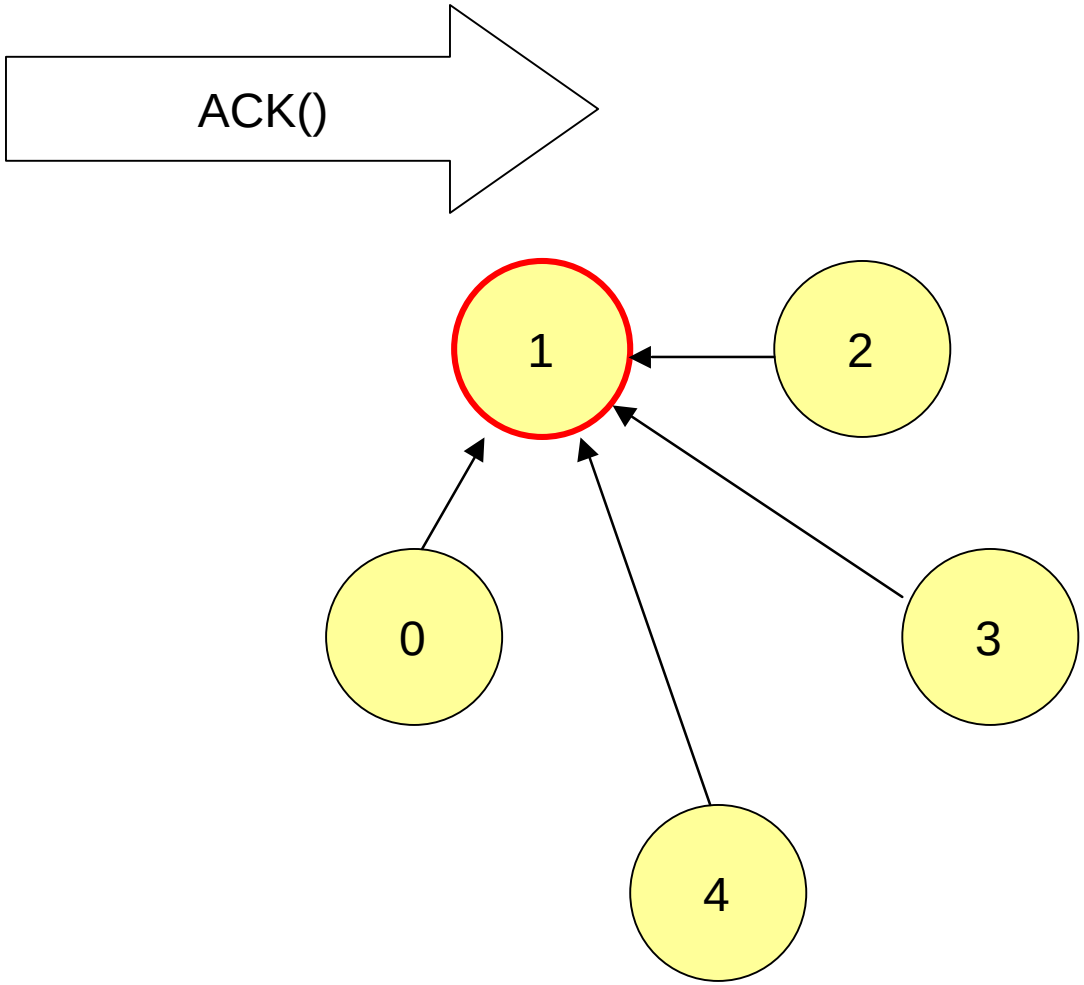
Paxos (6)

3. Phase: Falls ACCEPT von Mehrheit akzeptiert:
Wert verteilen



Paxos (7)

Erhalt des Ergebnisses bestätigen

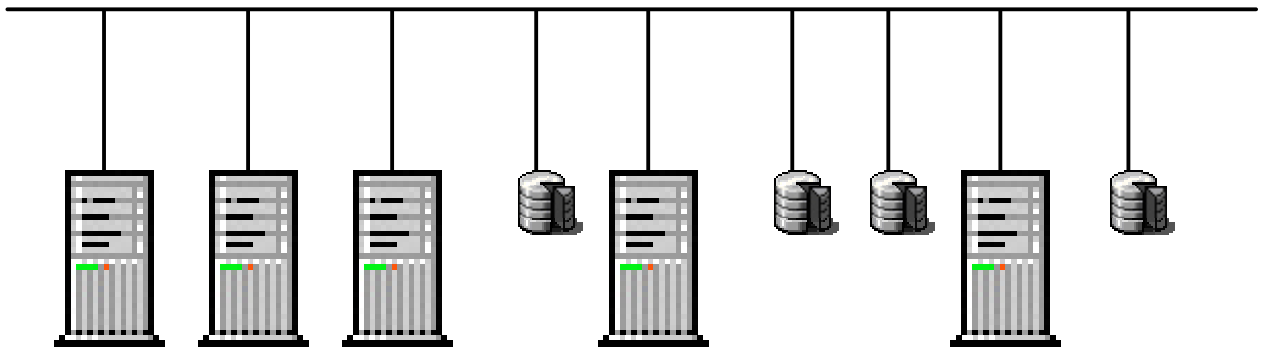


Basic Paxos => Multi Paxos

- Basic Paxos erreicht Einigung über nur einen Wert!
- Weiterentwicklung des Synod Protocols zum Parliamentary Protocol => Einigung auf ein Folge von Werten
- k Instanzen von Basic Paxos
- k wird in alle Nachrichten mit aufgenommen
- PREPARE(n,k) impliziert ein prepare für alle kleineren k
- Lücken werden mit Einigung auf einen wertlosem Wert aufgefüllt (“The ides of February is national olive day”)

Disk Paxos (1)

- Netzwerk von Prozessoren und Platten
- funktioniert, solange es mindestens einen funktionierenden Prozessor gibt, der eine Mehrheit der Platten lesen und schreiben kann



Disk Paxos (2)

- Modellierung als deterministischer, endlicher Automat der eine Folge von Befehlen abarbeitet
- Damit ist das Problem darauf reduziert, dass sich die Prozesse darauf einigen, welcher Befehl an welcher Stelle abgearbeitet werden soll.
- Erweiterung von Paxos: Jeder Prozessor erhält einen Block auf jeder der Platten zugeteilt. Dort legt er die Daten ab, die er normalerweisse im nichtflüchtigen Speicher ablegen würde. Die anderen Prozessoren können den Block auch lesen und somit sehen, wo die anderen Prozessoren stehen ohne mit ihnen Nachrichten auszutauschen

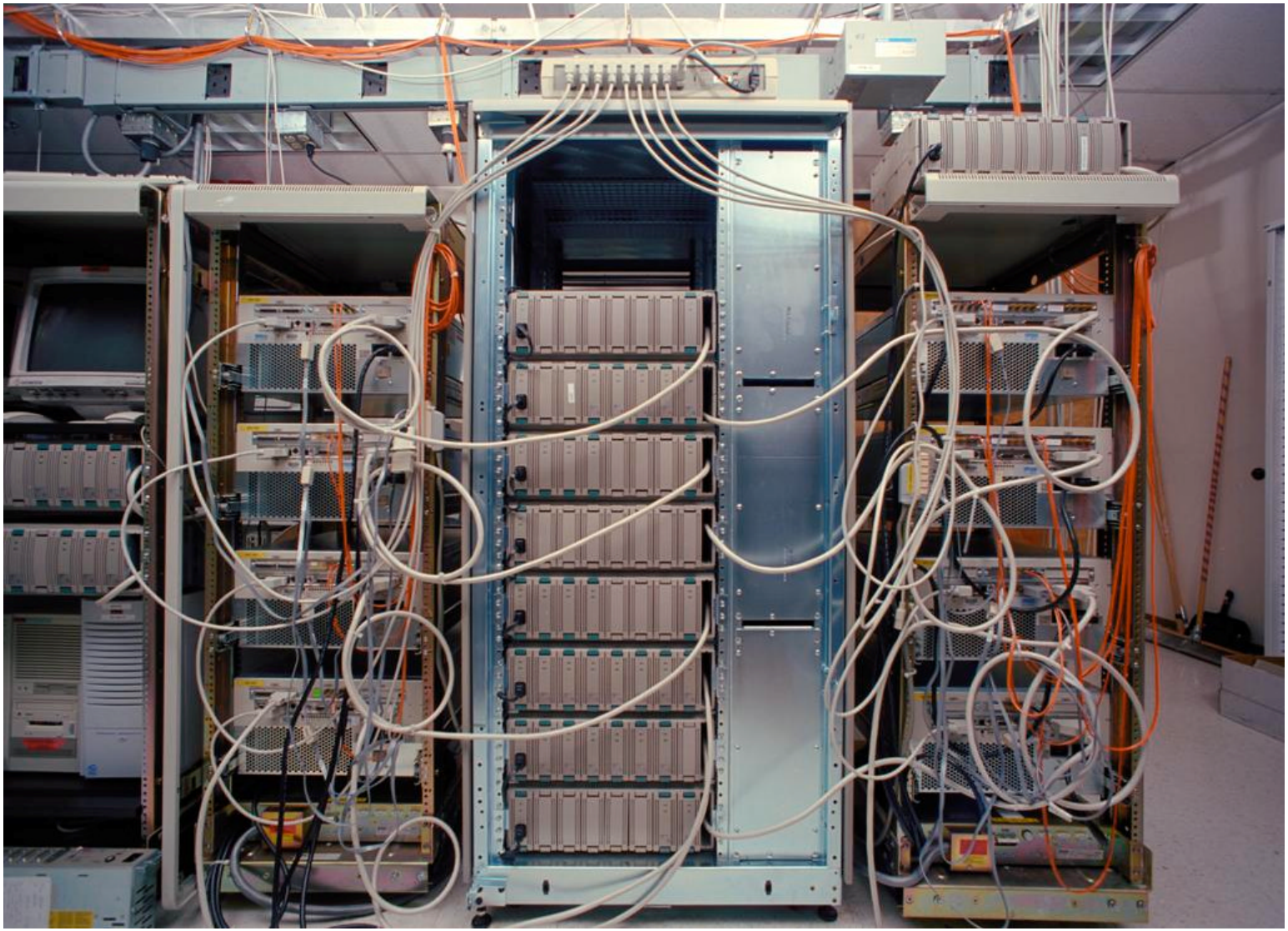
Petal (1)

- Compaq Research Labs
- Benutzt Paxos-Variante um Status-Informationen zu replizieren
- Erster Prototyp Januar 1996, zweiter Dezember 1996 - wurde als Speicher für Altavista benutzt
- Hervorragende Performance, entsprechend lokalem RAID
- Prototypen zeigten hervorragende Skalierbarkeit (fast linear)

Petal (2)


http://research.compaq.com/SRC/personal/eklee/Petal/petal.html

Petal (3)



Petal (4)

- Pläne für weitere Prototypen (mit je 100 Rechnern und Plattenarrays) die jedoch offenbar nie in die Tat umgesetzt wurden
- Frangipani: Filesystem das auf Petal aufsetzt