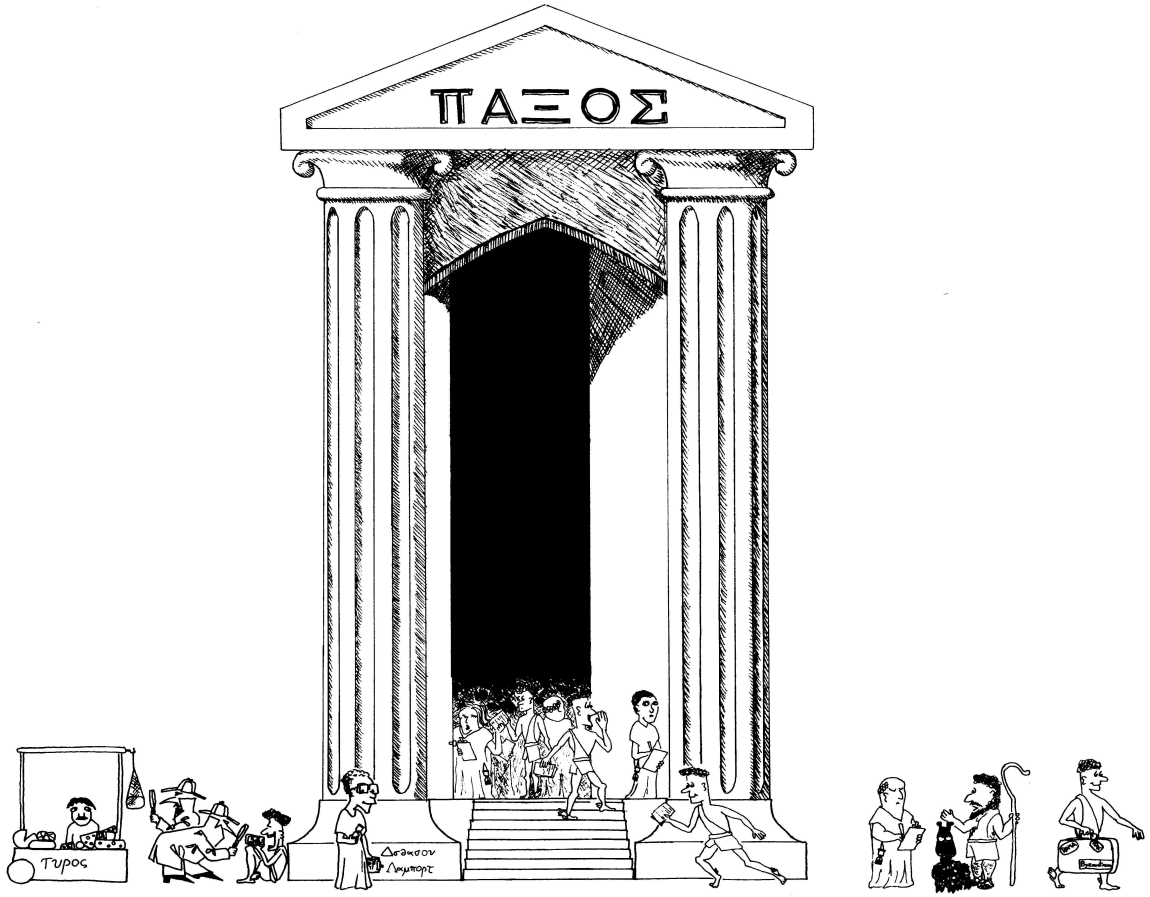
Das Teilzeitparlament

Yasin Taskin
10. Januar, 2003
Seminar
Verteilte Algorithmen



Inhalt

1. Einleitung
2. Basic Paxos
3. Disk Paxos
4. Praktische Anwendung von Paxos

1. Einleitung

Einleitung

- Geschichte
- Die Insel Paxos
- Eigenschaften der Inselbewohner
- Annahmen

Geschichte

- 1989 von Leslie Lamport entwickelt
- Ursprünglich als Erfindung die Anfang des letzten Jahrtausends auf der Insel Paxos gemacht wurde ausgegeben
- Fast keiner verstand seine Beschreibung
- Im Januar 1990 an ein Computermagazin zur Veröffentlichung gesandt, aber erst 1998 dort veröffentlicht

Die Insel Paxos

- Eine aufblühende Handelsinsel Anfang des letzten Jahrtausends
- Reichtum führte zur politischen Weiterentwicklung, so dass aus einer einstigen Theokratie eine Parlamentarische Regierung entstand
- Die Bewohner auf Paxos waren ihrem Leben sehr verbunden und wollten ihre Freizeit nicht nur im Parlament verbringen, also kamen und gingen sie wann sie wollten
- Problem: Suche nach einer Lösung für ein funktionierendes Teilzeitparlament für den Erlass von Gesetzen

Eigenschaften der Inselbewohner (1)

- Die Parlamentsmitglieder auf Paxos hatten keine Sekretäre, also führten sie alle ein Hauptbuch, eine Feder und ein kleines Fass Tinte mit sich, da sie sobald sie das Parlament verließen alles besprochene wieder vergaßen.
  - Diese Bücher enthielten niemals widersprüchliche Einträge.
  - Es gab keine Möglichkeit Einträge zu löschen, nur das durchstreichen war möglich.

- Paxons verließen manchmal in Scharen das Parlament um essen zu gehen, so dass es nicht mehr möglich war Abstimmungen durchzuführen, da eine Mehrheit von ihnen fehlte musste man warten bis sie wieder kamen.
  - Lösung: Eine Mehrheit der Mitglieder musste solange im Parlament bleiben bis eine Abstimmung beendet war.

Eigenschaften der Inselbewohner (2)

- Die Parlamentsmitglieder kannten sich alle, es konnten keine neuen Mitglieder im Parlament aufgenommen werden, da die neuen Mitglieder nicht gewusst hätten wie eine Mehrheit definiert ist.

- Einträge von Mitgliedern die das Parlament verließen mussten auch den anderen Mitglieder die später hinzukamen bekannt sein, Paxons waren bemüht das jeder wusste welche Gesetze Gültigkeit hatten.

Annahmen

- Die Akkustik im Parlamentsgebäude war so schlecht, das es Boten gab die Nachrichten zu jedem Mitglied überbrachten

- Die Legende besagt das die Boten und Parlamentarier so ehrlich waren das sie niemals Nachrichten verfälschten

- Die Boten durften allerdings:
  - Nachrichten vergessen
  - Nachrichten duplizieren
  - Sich viel Zeit lassen bei der Auslieferung der Nachrichten, manche fuhren sogar zwischenzeitlich in den Urlaub

2. Basic Paxos

Was macht der Algorithmus?

- Fehlertoleranter verteilter Algorithmus, indem Prozesse Werte vorschlagen können
- Einigung auf genau einen der vorgeschlagenen Werte

Der Algorithmus

- Grundmodell
- Fehlerannahmen
- Safety Requirements
- Funktionsweise des Basic-Protokoll
- Basic-Protokoll => Synod-Protokoll (Priesterversammlung)

Grundmodell

- Direkte asynchrone Kommunikation zwischen den Prozessen mittels Nachrichten
- Prozesse benötigen einen sicheren Speicher (Safe Storage), auf dem sie Werte ablegen können
  - Nach Ausfällen müssen sie auf ihre wichtigen Daten wieder Zugriff haben
- Mehrheit muss in irgendeiner Form festgelegt sein
  - In unserem Fall eine zahlenmäßige Mehrheit
- Alle beteiligten Prozesse sind bekannt

Fehlerannahmen

- Keine Byzantinischen Fehler
- Prozesse dürfen in beliebiger Anzahl und Häufigkeit ausfallen
- Wiederanlauf erlaubt
- Nachrichten dürfen:
  - verloren gehen
  - verzögert werden
  - dupliziert werden

Safety Requirements

1. Jede Abstimmung hat eine eindeutige Nummer.

2. Mehrheiten zweier Abstimmungen haben mindestens einen gemeinsamen Prozess.

3. Für jede Abstimmung B gilt:

   Wenn ein Prozess in B`s Mehrheit schon an einer früheren Abstimmung teilgenommen hat, dann ist der Wert der Abstimmung B gleich dem Wert der letzten Abstimmung in der ein Prozess gewählt hat.

Beispiel zu Safety Requirements

#	Wert	Stimmen				
2	$	A	B	C	D	
5	&	A	B	C		E
24	$		B		D	E
27	&	A		C	D	

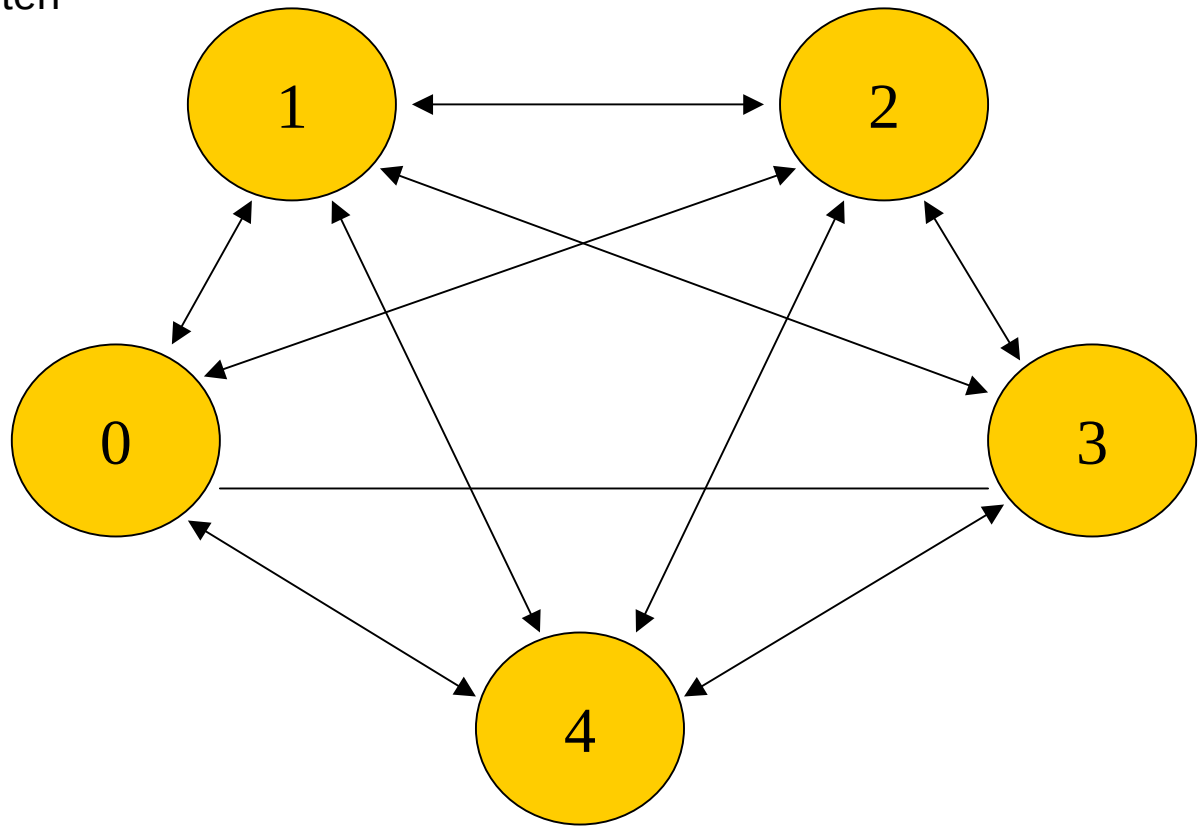
Funktionsweise des Basic-Protokolls

Der Algorithmus ist in ein 3 Phasen Protokoll eingeteilt:

1. Phase: Prozesse können Werte vorschlagen
2. Phase: Prozesse einigen sich auf einen der vorgeschlagenen Werte
3. Phase: Wenn ein Wert gewählt wurde, sollen alle Prozesse es erfahren

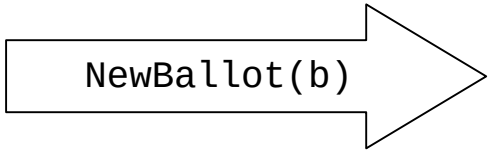
Basic-Protokoll

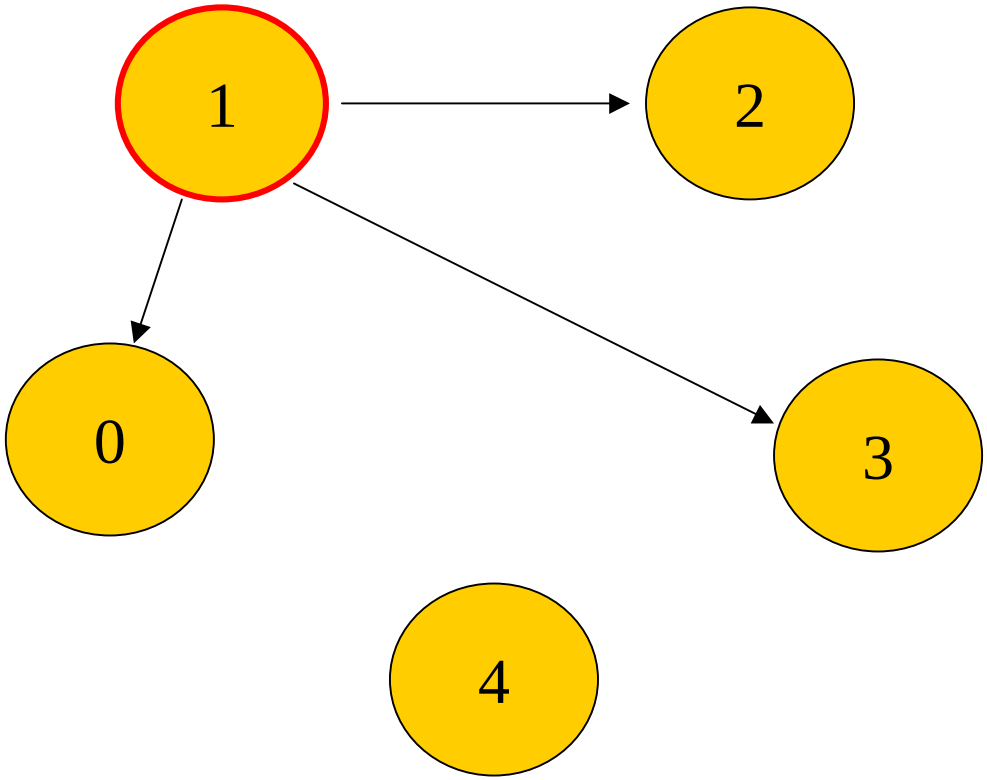
Jeder Prozess kann Vorschläge machen, akzeptieren und erfolgreiche Einigungen verbreiten



1.Phase (1)

„Vorbereitung“

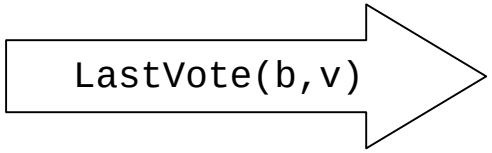


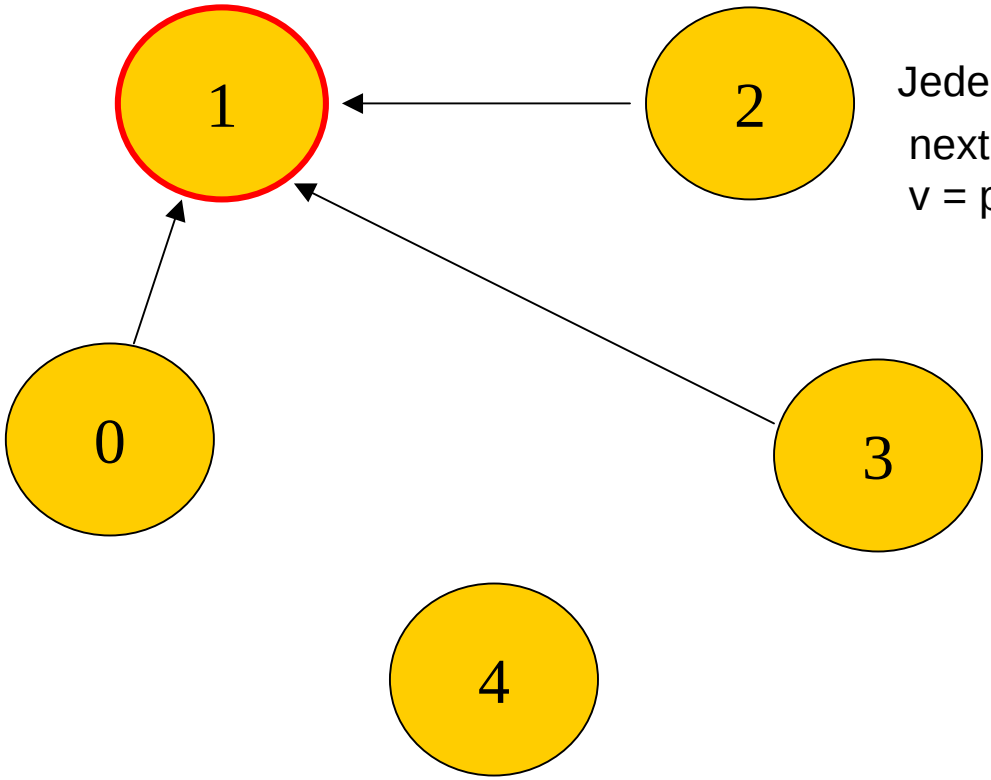


Prozess p setzt:
lastTried[q] = b

1.Phase (2)

Antwort auf NewBallot, falls b > nextBal[q]





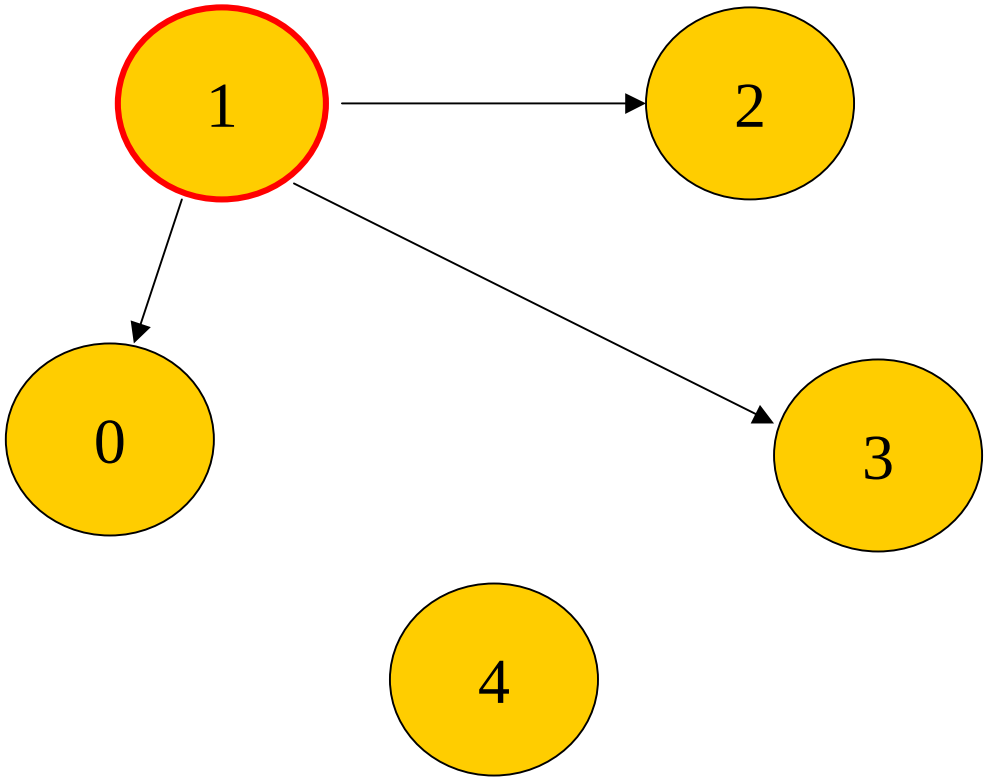
Jeder Prozess q merkt sich:

nextBal[q] = b und setzt
v = prevVote[q]

2.Phase (1)

Falls NewBallot von Mehrheit akzeptiert, beginnt die Abstimmung.

```
BeginBallot(b,d)
```

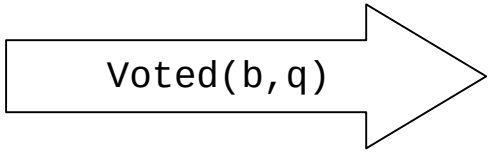


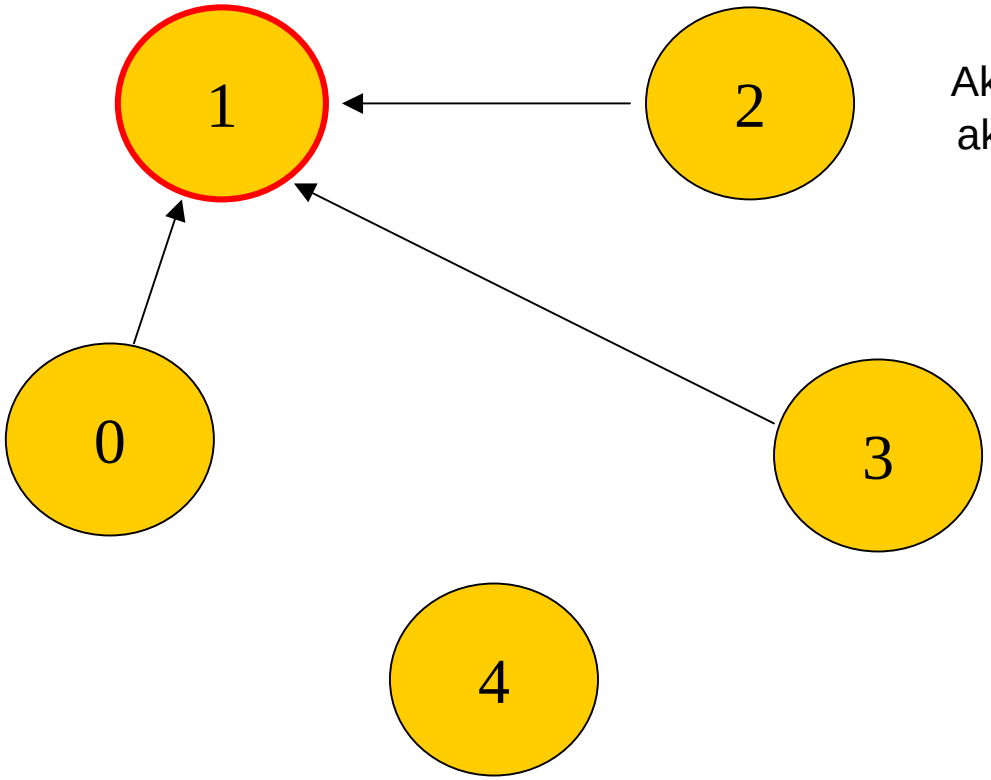
Prozess p überprüft:

b = lastTried[p]

2.Phase (2)

Antwort auf BeginBallot, falls b = nextBal[q]



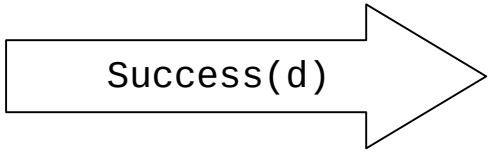


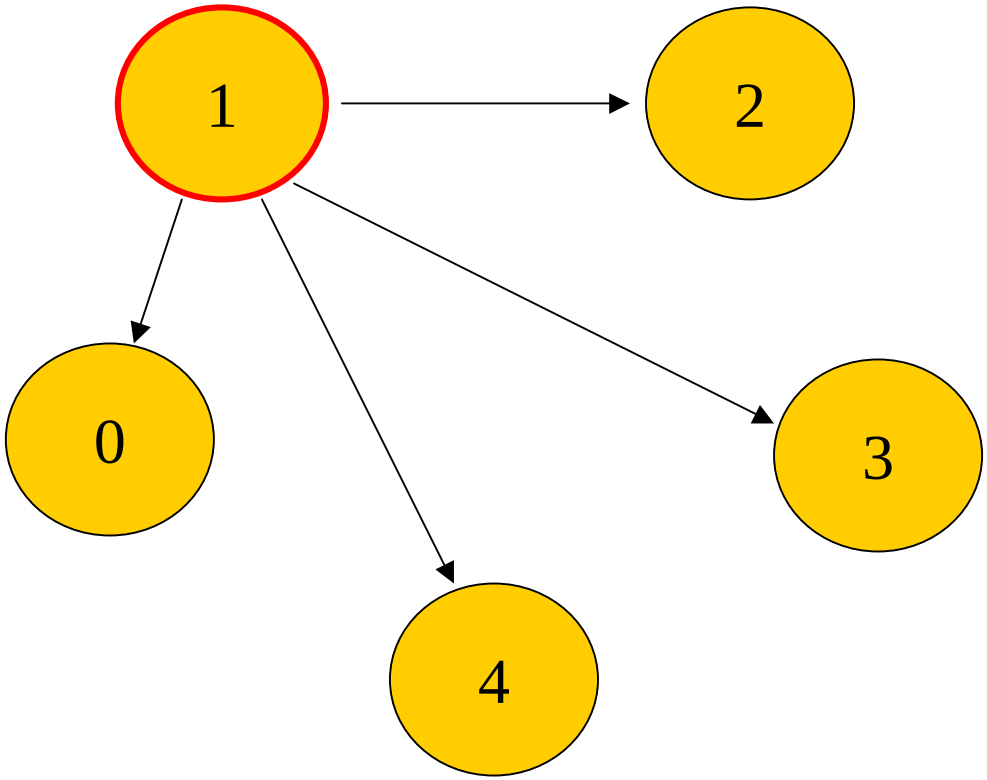
Akzeptierender Prozess q aktualisiert prevVote[q]

3.Phase (1)

Falls BeginBallot von Mehrheit akzeptiert:

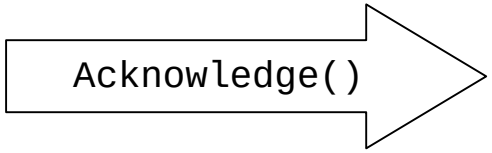
Wert sichern und verteilen an alle

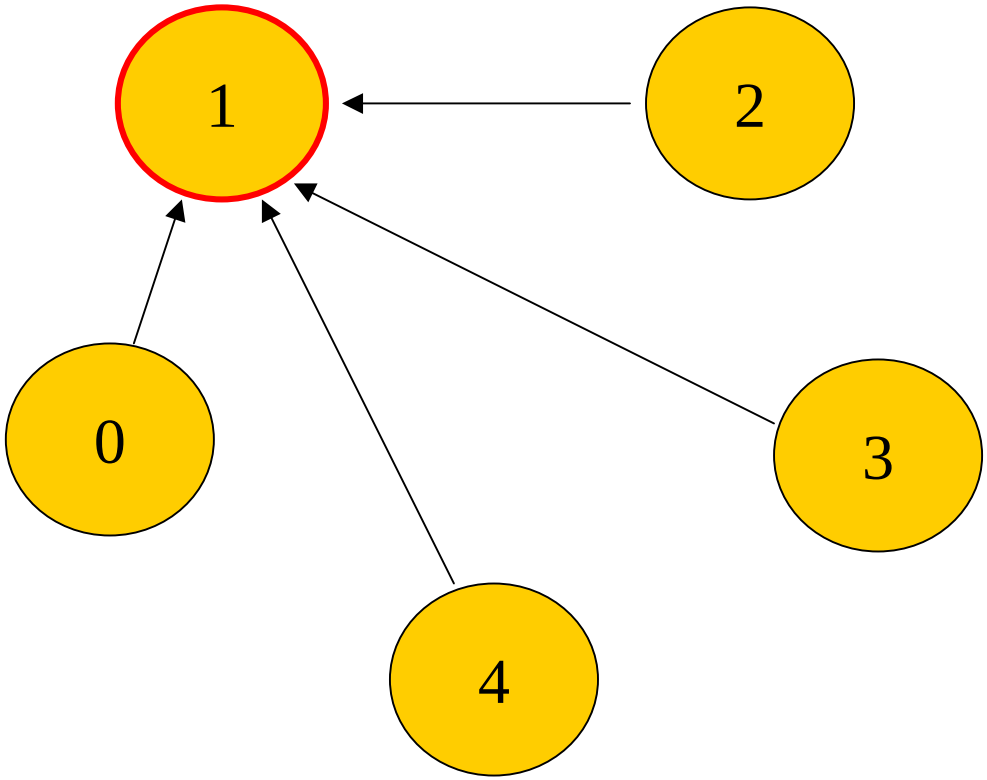




3.Phase (2)

Ergebnis sichern und erhalt bestätigen





Zusammenfassung

Mit dem Basic-Protokoll haben wir jetzt einen fehlertoleranten Algorithmus erhalten, der unter den angegeben Voraussetzungen die Safety Requirements erfüllt.

Die Konsistenz der Daten bleibt zwar auch erhalten, wenn die Prozesse gar nichts tun, aber das Ziel des Algorithmus ist es ja eigentlich eine Einigung zu erzielen => Synod-Protokoll

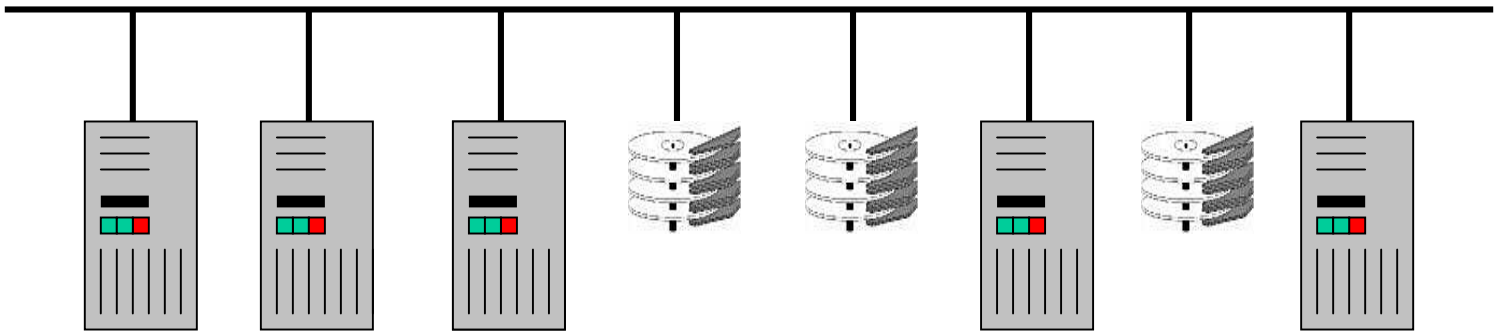
Basic-Protokoll => Synod-Protokoll

- Das Synod-Protokoll benutzt dieselben 3 Phasen wie das Basic-Protokoll
- Um Fortschritt zu optimieren müssen noch zusätzliche Bedingungen aufgestellt werden:
  - Es gibt einen eindeutigen Leader der die Abstimmungen durchführt
  - Eine Mehrzahl von Prozessen kann miteinander kommunizieren

3. Disk Paxos

Disk Paxos

- Verallgemeinerung des Paxos Algorithmus
- Netzwerk von Prozessoren und Platten
- Funktioniert, solange es mindestens einen funktionierenden Prozessor gibt, der eine Mehrheit der Platten lesen und schreiben kann.
- Jeder Prozessor erhält einen Block auf jeder der Platten zugeteilt. Dort legt er die Daten ab, die er normalerweise im sicheren Speicher ablegen würde.



Der Algorithmus

- Ein Prozessor führt eine Einigung in zwei Phasen aus:

  1.Phase: Es wird versucht ein Wert zu ermitteln

  2.Phase: Der ermittelte Wert wird zurückgeliefert und vom Prozessor ausgegeben

Vorbereitungen

Jeder Prozessor startet zu Beginn mit seinem eigenen Vorschlagswert *input[p].*

Um den Algorithmus auszuführen benötigen wir folgende Datenstruktur *dblock[p]:*

- *mbal* die momentane Abstimmungsnummer
- *bal* die größte Abstimmungsnummer für die *p* Phase 2 erreicht hat
- *inp* der Wert den *p* in Abstimmungsnummer bal versucht zurückzuliefern

Die Komponenten *mbal*, *bal* und *inp* sind dieselben wie *nextBal[p], lastTried[p]* und *prevVote[p]* des Synod-Protokolls.

disk[d][p] sei der Block der für Prozessor *p* auf der Platte *d* zugeteilt ist, auf die er schreiben kann.

Initial ist *bal 0* und *inp* ein spezieller Zustand *„kein Wert“.*

Funktionsweise des Disk-Paxos (1)

1.Phase:

- Zunächst wird versucht *dblock[p]* auf *disk[d][p]* zu schreiben und dann *disk[d][q]* für alle anderen Prozessoren *q* zu lesen.
- Durch das lesen der Informationen der anderen Prozessoren kann ein Prozessor *p* selbst feststellen ob seine Wahl Gültigkeit besitzt.
- Abbruch der Wahl falls, *disk[d][q].mbal* > *dblock[p].mbal*, also ein Prozessor eine Abstimmung besitzt mit einer höheren Abstimmungsnummer.
- Die 1.Phase ist beendet wenn Prozessor *p* eine Mehrheit von Platten beschrieben und gelesen hat.

Funktionsweise des Disk-Paxos (2)

2.Phase:

- Prozessor *p* wählt einen Wert für *dblock[p].inp*:
  - Falls alle anderen Prozessoren *q* nur Initialwerte in den gelesenen *disk[d][q].inp* haben, so benutzt *p* seinen *input[p]* Wert.
  - Sonst benutzt *p* den Wert *disk[d][q].inp* für den *disk[d][q].bal* am größten ist.
  - Diesen gewählten Wert versucht Prozessor *p* dann zurückzuliefern.

Disk Paxos / Basic Paxos

- Die Komponenten *mbal, bal* und *inp* sind dieselben wie *nextBal[p] lastTried[p]* und *prevVote[p]* des Synod-Protokolls.
- Phase 1 entspricht dem senden von *NextBallot* und empfangen von *LastVote* Nachrichten im Synod-Protokoll.
- Phase 2 entspricht der *BeginBallot* und das empfangen von *Voted* Nachrichten.

Unterschiede Disk Paxos / Basic Paxos

- Das Synod-Protokoll sendet bei seiner *NextBallot* Nachricht nur den *mbal* Wert, nicht die Werte *bal* und *inp*, um das Synod-Protokoll zu erhalten müsste man den Algorithmus von Disk Paxos so verändern das er in der 1.Phase nur den *mbal* Wert auf seinen Plattenblock schreibt und die Felder von *bal* und *inp* unverändert lässt.
  - Die Veränderung würden es erschweren den Algorithmus auf echten Platten zu realisieren

4. Praktische Anwendung von Paxos

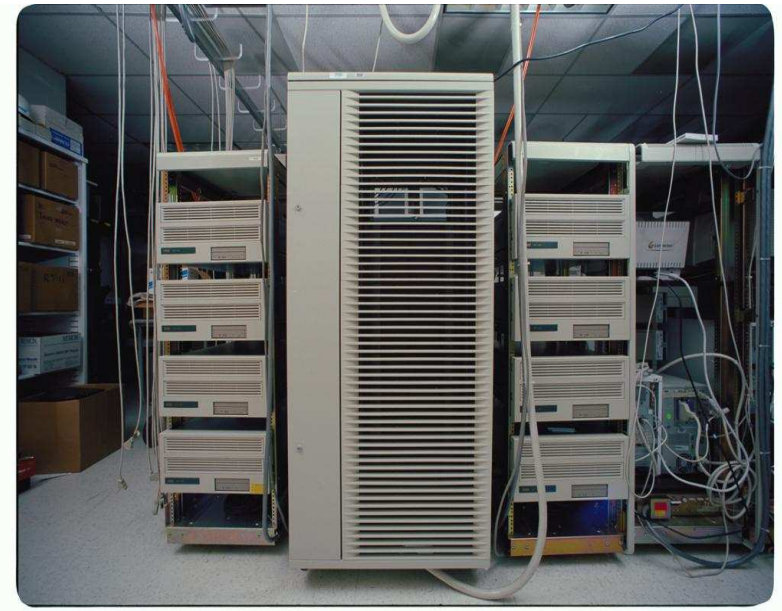
Petal (1)

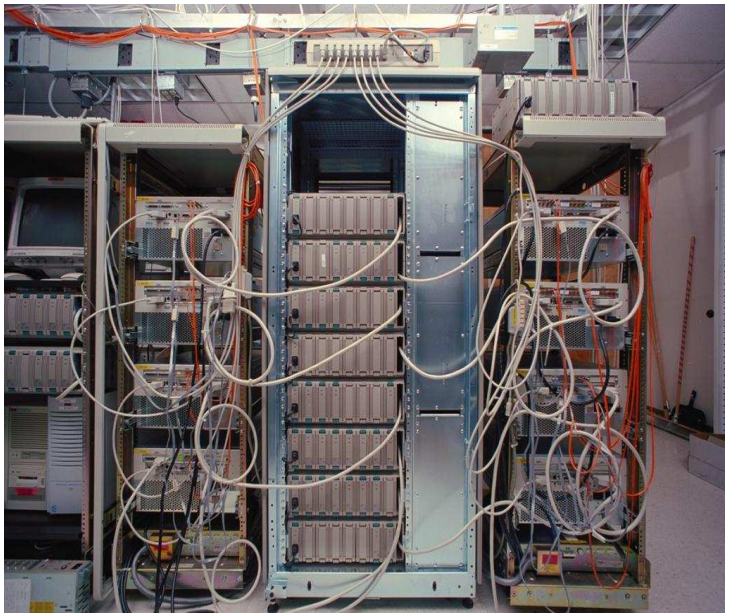
Bei Petal handelt es sich um ein System von über ein (möglichst schnelles) Netzwerk miteinander verbundenen Servern, die gemeinsam eine Menge physikalischer Disks verwalten.

- 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

Petal (2)

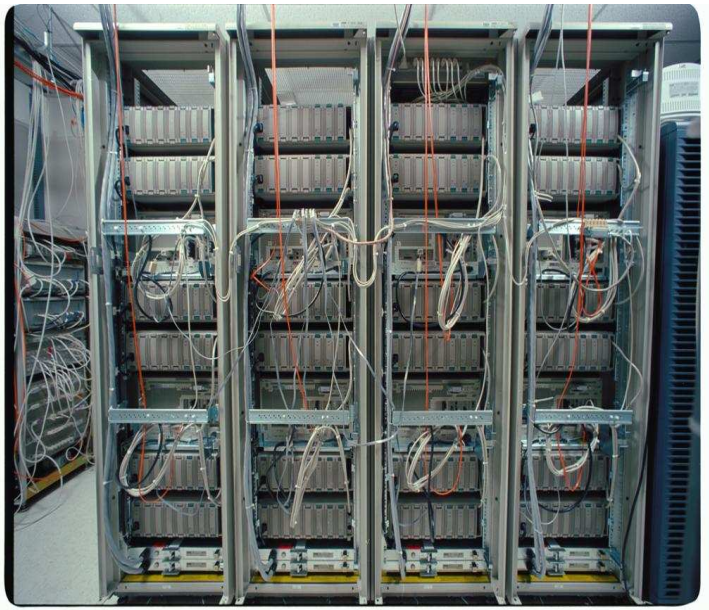
Erster Petal Prototyp





Petal (3)

Zweiter Petal Prototyp



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

Literaturverzeichnis

Leslie Lamport:
Part-Time Parliament, 1989

Leslie Lamport:
Paxos made simple, 2001

Eli Gafny, Leslie Lamport:
Disk Paxos, 2002

Michael Meier:
Fehlertolerante, konsistente Replikation mit Hilfe des Paxos Algorithmus, 2002