

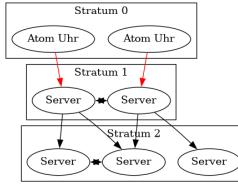




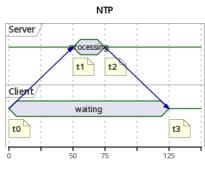
## Übung Cristians Algorithmus

- Laufzeit Nachricht  $P \rightarrow S$  und  $S \rightarrow P$  jeweils 100 ms.
- $S$  benötigt 10 ms für die Bearbeitung der Anfrage.
- $t = 500$  ms.
- Welche Zeit wird bei  $P$  eingestellt?

## NTP (Network Time Protocol): Diagramm



## NTP Berechnung



- Client startet Anfrage zu  $t_0$ .
- Server empfängt Anfrage zu  $t_1$  und sendet Antwort zu  $t_2$ .
- Client empfängt Antwort zu  $t_3$ .
- $t_0$  und  $t_3$  jeweils in Client-Zeit,  $t_1$  und  $t_2$  in Server-Zeit.
- Offset berechnet sich:  $offset = \frac{(t_3 - t_0) + (t_2 - t_1)}{2}$
- Offset wird verwendet, um die Zeit graduell anzupassen.

## NTP graduelle Anpassung

- Was passiert bei  $offset < 0$ ?
- Uhren sollten nicht rückwärts laufen!
- Bsp: jeder tick erhöht Software Uhr um 10 ms.
  - Idee: Verringerung des Inkrement, um Uhr schrittweise anzugelichen.
- Wird auch verwendet, um die Uhr vorwärts anzupassen.
- Graduelle Anpassung wird bei offsets > 128 ms nicht verwendet.



## NTP Übung

- Client stellt Anfrage bei  $t_0 = 100$  ms.
- Server empfängt Anfrage bei  $t_1 = 50$  ms und benötigt 10 ms zur Bearbeitung.
- Berechne das offset bei:
  - $t_{C \rightarrow S} = t_{S \rightarrow C} = 5$  ms
  - $t_{C \rightarrow S} = 5$  ms und  $t_{S \rightarrow C} = 10$  ms
- Auf welche Zeit wird die Uhr des Client jeweils gestellt?

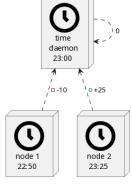
## NTP

- Neben dem offset wird noch das delay berechnet:  
 $delay = (t_3 - t_0) - (t_2 - t_1)$
- Es werden 8 offset-delay Paare ermittelt und das Paar mit dem geringsten delay verwendet.
- NTP erreicht Genauigkeiten von 1-50 ms.
- 30ms Verzögerung ist die Obergrenze für Interaktives!



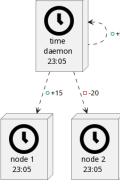
## Berkeley Algorithmus - Schritt 2

Maschinen antworten mit ihrem offset.



## Berkeley Algorithmus - Schritt 3

Zeitserver berechnet Durchschnitt der Uhrzeiten und sendet offsets an Maschinen.



## Berkeley Algorithmus - Beobachtungen

- Bietet interne Synchronisation.
- System wird nicht mit externer Uhr (UTC) synchronisiert.
- Solange das System intern einen einheitlichen Zeitbegriff verwendet, können Operationen geordnet werden.

Clusterzeit.

## Spanner

- Nutzt timestamps in der Form  $[T_{lower} T_{upper}]^1$
- „True Time Service“
- Kommt auf 6ms Genauigkeit
- Transaktionen werden verzögert, damit  $T_{upper}$  auf jeden Fall verstrichen ist.

- Uhren sind immer mit einem Fehler versehen.



## Zusammenfassung Uhren + Synchronisation

- wall time vs. monotonic clocks
- Uhren leiden unter skew und drift.
  - $\rightarrow$  müssen synchronisiert werden.
- externe Synchronisation: Cristians Algorithmus, NTP
- interne Synchronisation: Berkeley
- Uhren haben Unsicherheit

## PAUSE

--- PAUSE ---



## NTP (Network Time Protocol): Ablauf

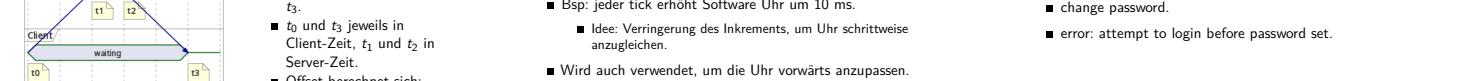


- Bestandteile des Systems werden in Strata unterteilt.
- Referenz-Uhren befinden sich in Stratum 0.
- Ein Server in Stratum  $n$  kontaktiert Server in Stratum  $n-1$  zur Synchronisation.
- Oft werden mehrere Server angefragt und die Ergebnisse statistisch behandelt (Mittel, Ausreißer).
- Durch innerhalb eines Stratum wird kommuniziert



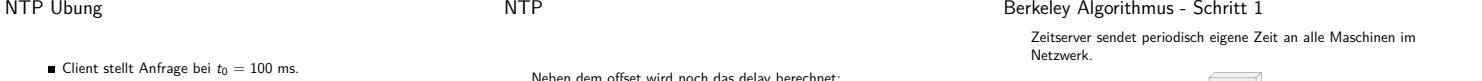
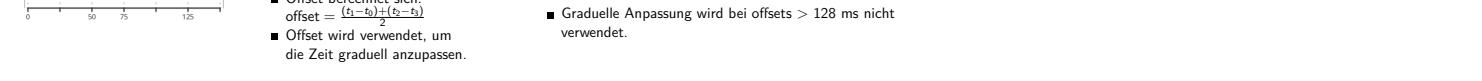
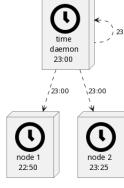
## Niemals rückwärts! Erfahrung

- change password.
- error: attempt to login before password set.



## Berkeley Algorithmus - Schritt 1

Zeitserver sendet periodisch eigene Zeit an alle Maschinen im Netzwerk.

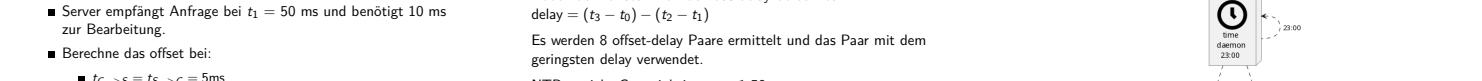


## Berkeley Algorithmus - Übung

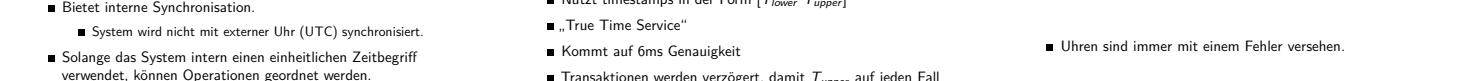
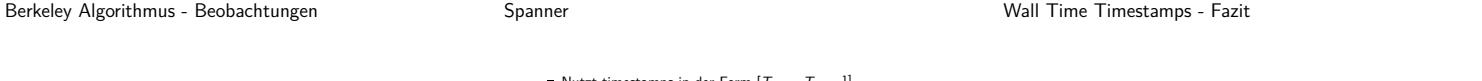
3 Rechner und ein Server im System.

Gebe die Nachrichten des Berkley Algorithmus an. Welche Zeit wird im System eingestellt?

| Nr  | von    | an     | Inhalt |
|-----|--------|--------|--------|
| 1   | Server | A,B,C  | 11:00  |
| 2   | Server | Server | 0      |
| ... |        |        |        |
| 9   |        |        |        |



## Wall Time Timestamps - Fazit



## Logische Uhren + Multicast ordnen

Ziele:

- Sie kennen Alternativen zu wall time clocks.
- Sie verstehen den Unterschied zwischen partieller und totaler Ordnung.
- Sie verstehen Lamport clocks.
- Sie kennen total geordneten Multicast mit Lamport clocks.
- Sie verstehen Vector clocks.
- Sie kennen kausal geordneten Multicast mit vector clocks.





## Lamport Uhren

- Die exakte Uhrzeit interessiert uns nicht
- Reihenfolge von Ereignissen
- ⇒ Zeit-Ordnung
- Timestamps sollen Kausalität berücksichtigen:
  - Wenn a kausal vor b passiert ist, dann  $\text{timestamp}(a) < \text{timestamp}(b)$ .
  - Tür wird geöffnet bevor man eintritt  $\Rightarrow$  dann ist  $\text{timestamp}(\text{Schritt}) > \text{timestamp}(\text{öffnen})$



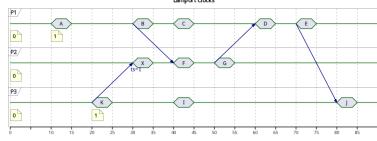
## Umsetzung Lamport Clocks

Jeder Prozess  $P_i$  erstellt einen lokalen Zähler  $C_i$  und wendet folgende Regeln an:

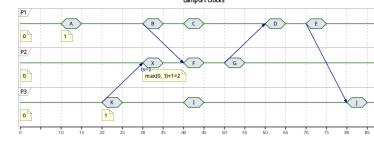
- Für 2 **sukzessive Ereignisse**, die in  $P_i$  stattfinden, wird  $C_i$  um 1 erhöht.
- Wenn eine Nachricht von  $P_i$  **gesendet** wird, erhält sie den timestamp  $\text{ts}(m) = C_i$ .
- Wenn eine Nachricht von  $P_j$  **empfangen** wird, setzt  $P_i$   $C_j$  auf  $\max(C_j, \text{ts}(m)) + 1$



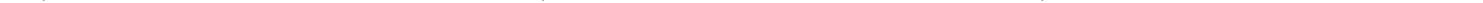
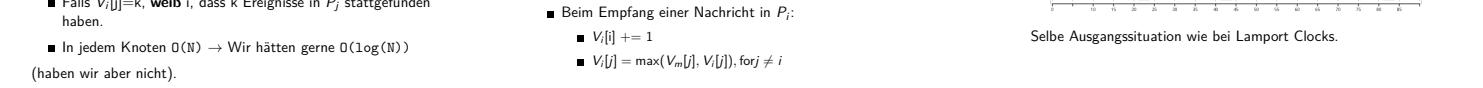
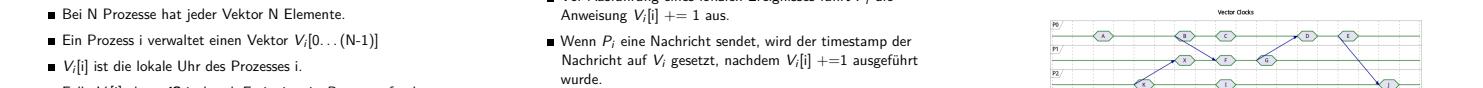
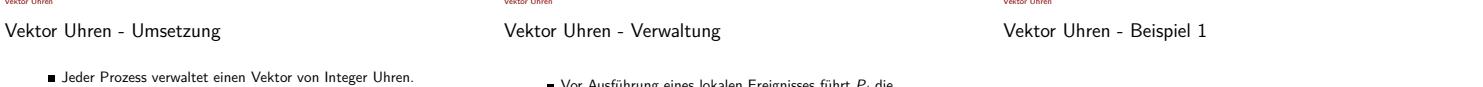
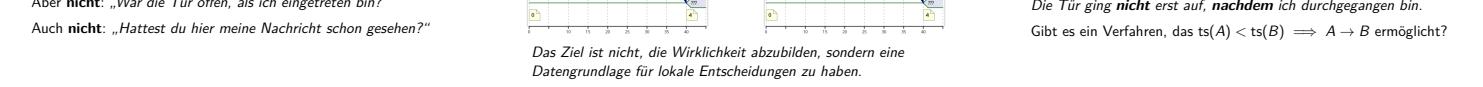
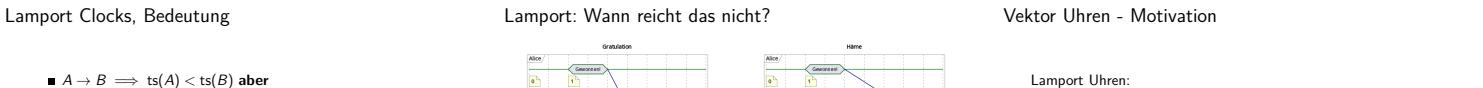
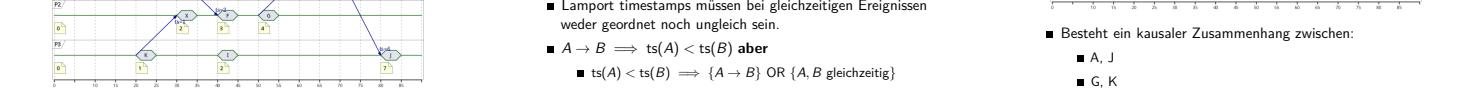
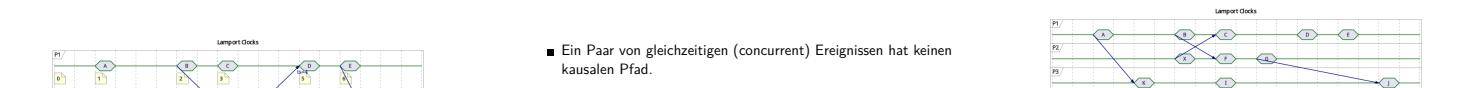
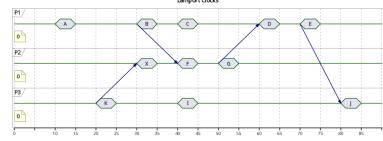
## Lamport Clocks - Schritt 2



## Lamport Clocks - Schritt 3

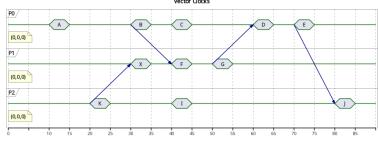


## Lamport Clocks - Schritt 4

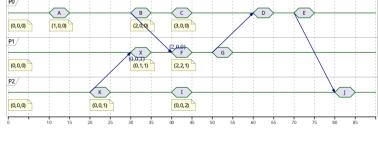




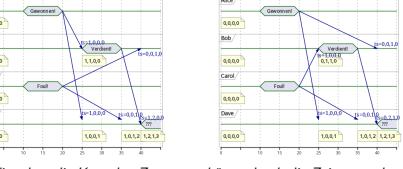
## Vektor Uhren - Beispiel 2



## Vektor Uhren - Beispiel 5



## Vektor Uhren - Beispiel: Lob oder Häme?



Wir sehen die Kausalen Zusammenhänge durch die Zeitstempel.



## Vektor Uhren - Abschluss

### Lamport Clocks:

- $A \rightarrow B \implies ts(A) < ts(B)$  aber
- $ts(A) < ts(B) \implies \{A \rightarrow B\}$  OR  $\{A, B$  gleichzeitig

### Vector Clocks:

- $ts(A) < ts(B) \implies A \rightarrow B$
- $\neg(ts(A) < ts(B)) \wedge \neg(ts(B) < ts(A)) \implies A, B$  gleichzeitig



## Kausal geordneter Multicast mit Vector Clocks

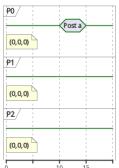
Mit wenigen Änderungen können Vector Clocks genutzt werden, um kausal geordnete Nachrichten sicherzustellen.

### Folgendes Beispiel:

- Ein Bulletin Board Service.
- Benutzer treten Gruppen bei.
- Posts werden an alle Gruppenmitglieder gemulticastet.
- Könnte vollständig geordnete Multicast verwenden.
- Wird aber nicht benötigt.



## Kausal geordneter Multicast - Beispiel 1



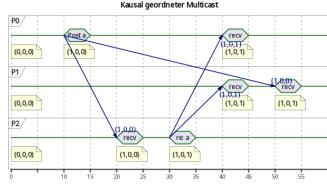
### Arne Babenhausrede

#### Verteilte Systeme 2: Koordination





## Kausal geordneter Multicast - Beispiel 4



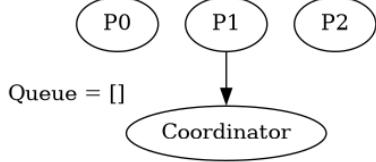
## Gegenseitiger Ausschluss

Ziele:

- Sie kennen die Grundlegenden Konzepte für Gegenseitigen Ausschluss.



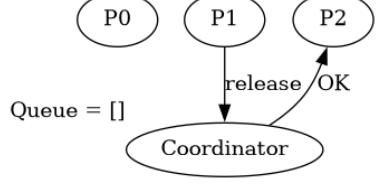
## Zentralisierter Algorithmus - Happy Path



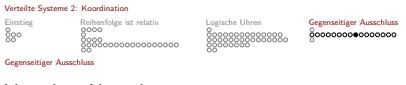
Falls die Ressource verfügbar ist, erhält der anfragende Prozess die Berechtigung.



## Zentralisierter Algorithmus - Ressource wird freie



Sobald die Ressource wieder frei wird, gibt der Koordinator die Ressource an den ersten Prozess in der Queue.



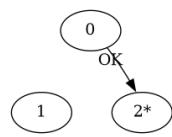
## Verteilter Algorithmus

- Wenn ein Prozess, die Ressource benötigt sendet er eine Nachricht an alle Prozesse (auch sich selbst).
- Bei Erhalt so einer Nachricht:
  - Prozess hält Ressource nicht und möchte sie nicht: sendet OK.
  - Prozess hält Ressource: antwortet nicht.
  - Prozess möchte Ressource: vergleiche timestamp der Nachricht mit timestamp der eigenen Nachricht. Der niedrigere timestamp gewinnt.
- Prozess wartet Antworten aller Prozesse ab. Sobald er sämtliche OKs erhalten hat, verwendet er die Ressource.

Voraussetzung: *Totale Ordnung der Nachrichten.*



## Verteilter Algorithmus - Gleichzeitiger Zugriff 3



- P0 benötigt die Ressource nicht mehr.
- P0 sendet OK an erste Anfrage in Queue.
- P2 erhält Zugriff.





## Token Ring Algorithmus - Bewertung

- Was passiert wenn ein Knoten ausfällt?
- Wieviele Nachrichten werden benötigt?

## Vergleich

| Algorithmus   | Nachrichten pro Ein/Austritt |
|---------------|------------------------------|
| Zentralisiert | 3                            |
| Verteilt      | $2N-1$                       |
| Token Ring    | $1, \dots, \infty$           |

## Zusammenfassung Gegenseitiger Ausschluss

- Mutex mit Koordinator
- Mutex verteilt
- Mutex Token Ring



## Zusammenfassung

### Reale Uhren:

- wall time vs. monotonic clocks
- Skew and Drift
- Synchronizing: extern (Cristian, NTP), intern (Berkeley)

### Logische Uhren:

- Lamport: Ein Zähler pro Knoten. „Wenn es vorher war, dann ist der Zeitstempel kleiner.“
- Vektor: N Zähler in jedem der N Knoten. Kausalität. „Wenn der Zeitstempel kleiner ist, dann war es vorher.“

Ausschluss: Koordinator oder verteilt  $\Rightarrow$  Zusätzliche Nachrichten.



## Für koordinierte Projekte!

## Zusammenfassung

### Arne Babenhausen

### Verteilte Systeme 2: Koordination

### Literatur

Bilder:

### Arne Babenhausen

### Verteilte Systeme 2: Koordination

### Arne Babenhausen

### Verteilte Systeme 2: Koordination