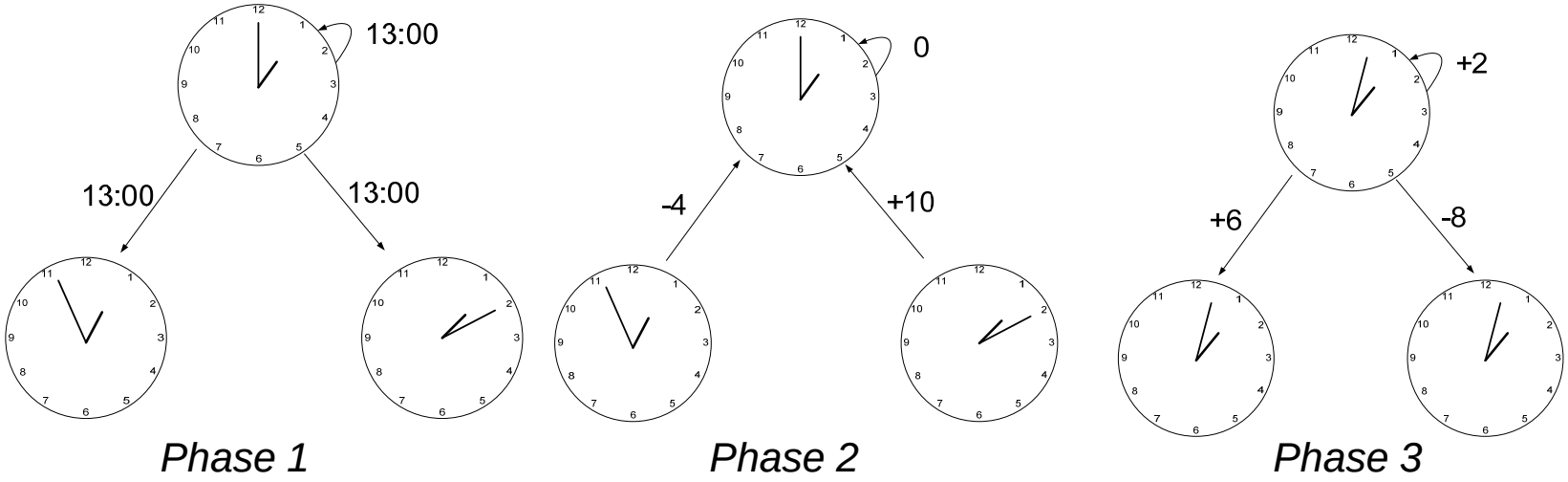
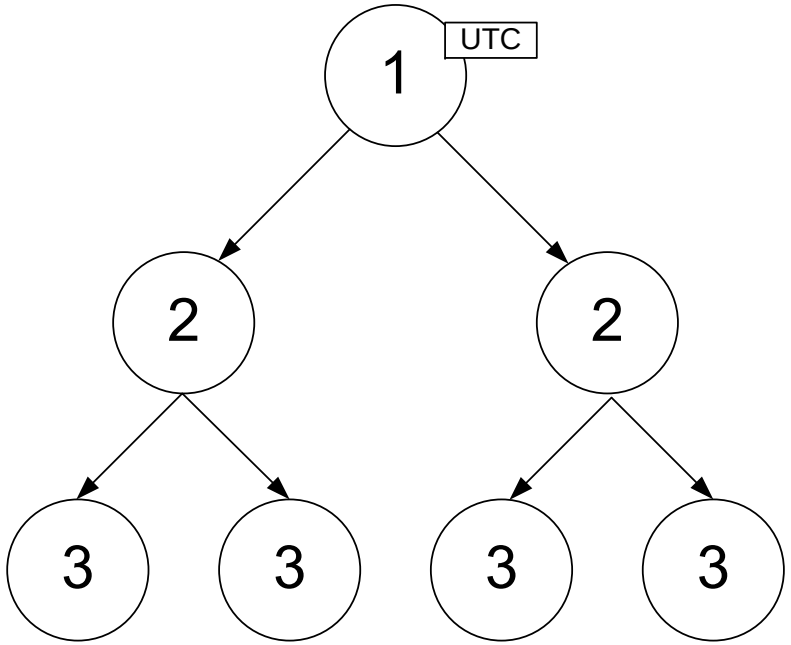
Algorithmus von Berkeley (1989)

- Annahme: kein UTC-Empfänger verfügbar
- Algorithmus (zentral, intern):
  - ein Rechner agiert als aktiver Time-Server.
  - Der Server fragt periodisch die Zeiten/Unterschiede aller anderen Rechner ab (Phase 1) und ermittelt den Durchschnittswert (Phase2).
  - In Phase 3 wird der errechnete Wert an alle anderen Uhren ausgegeben.



NTP: Network Time Protocol (1982)

- Problem: Die angegebenen Algorithmen funktionieren nur in kleinen statischen Netzen.
- Das NTP Protokoll bietet eine Möglichkeit in großen Netzen eine Synchronisation zu gewährleisten.
- Die Netze können dabei dynamisch konfiguriert werden, um eine zuverlässige Synchronisation zu gewährleisten.
- Die Grundstruktur von NTP ist ein hierarchisches Modell (mit verschiedenen Strata/Schichten).
  - Der Dienst wird durch ein verteiltes Serversystem geleistet.
  - Primäre Server sind direkt mit einer UTC-Quelle verbunden.
  - Sekundäre Server synchronisieren sich mit primären Servern usw.
  - Jede zusätzliche Schicht verursacht einen zusätzlichen Zeitversatz von 10-100ms.



NTP Algorithmus (schematisch)

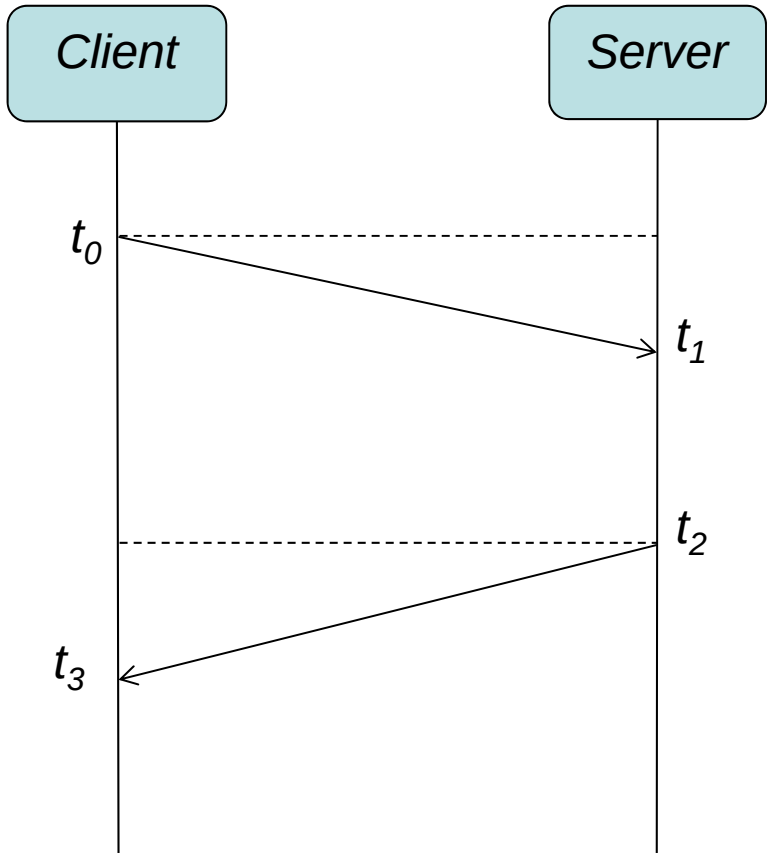
- Synchronisation besteht aus zwei Teilen:
  - Offset der Uhren

$$\frac{(t_1-t_0) + (t_2-t_3)}{2}$$

  - Round-trip delay

$$(t_3-t_0) - (t_2-t_1)$$

- Voraussetzung für präzise Synchronisation: Ein- und ausgehende Routen haben symmetrisches Delay.



Uhren und Synchronisation

Synchronisation bei fehlerbehafteten Uhren

Problemstellung

- Die bisherigen Algorithmen basierten alle auf der Annahme von fehlerfreien Uhren.
- Im Folgenden werden Algorithmen betrachtet, die mit einer maximalen Anzahl von ***m*** fehlerbehafteten Uhren umgehen können.
- Insgesamt soll das System aus ***n*** Uhren bestehen. Betrachtet werden im Besonderen auch byzantinische Fehler (die fehlerhafte Einheit kann beliebige Ausgaben produzieren).
- Die maximal zulässige Abweichung zweier Uhren bezeichnen wir mit ε.
- In Frage kommen dabei nur verteilte Algorithmen, um einen Single-Point-of-Failure auszuschließen.

Konvergenzalgorithmus (Leslie Lamport, 1985) [1]

- Algorithmus:
  - Jede Einheit liest die Uhr der anderen Rechner und berechnet den Mittelwert.
  - Ist die Abweichung einer Uhr größer als ε, so verwendet der Algorithmus stattdessen den Wert der eigenen Uhr.
- Aussage:
  - Der Algorithmus arbeitet erfolgreich, falls gilt: $n \geq 3m$.
- Annahmen:
  - vernachlässigbare Ausführungszeit
  - Einheiten lesen zeitgleich die Uhren ab bzw. Unterschiede sind vernachlässigbar

[1] Synchronizing Clocks in the Presence of Faults, Leslie Lamport and P.M. Melliar-Smith, SRI International, Menlo Park, California

Konvergenzalgorithmus (Leslie Lamport, 1985)

- Beweis:
  - Seien *p*,*q* zwei fehlerfreie Einheiten, *r* eine beliebige Einheit.
  - Sei *t(p,r)* die Uhrzeit von r, die die Einheit *p* für die Mittelwertsberechnung verwendet.

    $\Rightarrow$ r fehlerfrei: $t(p,r) \approx t(q,r)$

    $\Rightarrow$ r fehlerbehaftet $|t(p,r)-t(q,r)|<3\varepsilon$
  - Einheit p stellt seine Uhr auf: $1/n * \sum_r t(p,r)$
  - Einheit q stellt seine Uhr auf: $1/n * \sum_r t(q,r)$
  - Schlechtester Fall:
    - (n-m) Uhren fehlerfrei: $t(p,r) \approx t(q,r)$
    - m Uhren fehlerbehaftet $|t(p,r)-t(q,r)|<3\varepsilon$

$\Rightarrow$ Differenz beider Uhren: $\Delta(p,q)=1/n\ |\sum_r t(p,r)-\sum_r t(q,r)| \leq m/n*3\varepsilon < \varepsilon$

Kapitel 3

Modellgetriebene Entwicklung von Echtzeitsystemen (inkl. Werkzeuge)