Verteilte Systeme

9. Verteilte Algorithmen

Bereits behandelte Bereiche

- Logische Uhren
  - Keine globale Uhrensynchronisation möglich (Theorie)
  - Kausalitätserhaltender Zeitbegriff
- Multicast
  - Überflutungs- und Echo-Algorithmen
  - Ordnungs- und Zuverlässigkeitsgrad
- Wechselseitiger Ausschluß
  - Symmetriegrad einer Lösung
  - Fairneß
- Verteilte Terminierung
  - Kommunikationsorientierte Terminierung
  - Ergebnisorientierte Terminierung

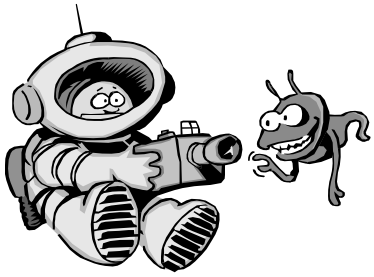
Weitere Bereiche

- Schnappschußalgorithmen
  - Kein atomarer, globaler Zustand direkt abfragbar (Theorie)
  - Abschwächung: konsistente Schnappschüsse
  - Grundlage z.B. für ergebnisorientierte Terminierung
- Election
  - Wahl eines Prozesses aus mehreren Kandidaten
  - Fairness nicht notwendig
- ...

- Allgemein: Agreement-Protokolle
  - Alle Beteiligten einigen sich auf etwas
  - Verteilte Algorithmen sind häufig Spezialfälle

Verteilte Systeme, Sommersemester 1999 Folie 9.3

Schnappschußalgorithmen

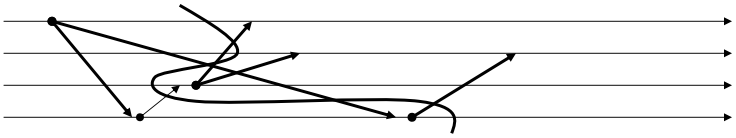


- Verteilter Zustand
- Schnappschuß eines globalen Zustands
  - Lokale Prozeßzustände
  - Unterwegs befindliche Nachrichten
- Breites Anwendungsfeld
  - Verteilte Terminierung
  - Verklemmung
  - Wechselseitiger Ausschluß
  - ...
- Nachteile
  - Meist hohes Nachrichtenaufkommen
  - Nicht „Verteilt" genug

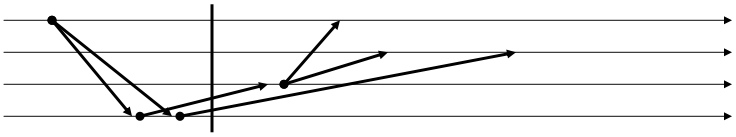
Verteilte Systeme, Sommersemester 1999 Folie 9.4

Schnitte im Zeitdiagramm

- Abfrage erreicht nicht alle Prozess gleichzeitig



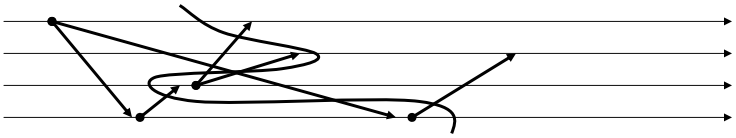
- Äquivalente Berechnung



  - Gumminbandtransformation

Inkonsistente und konsistente Schnitte

- Inkonsistent: Nachrichten aus der Zukunft



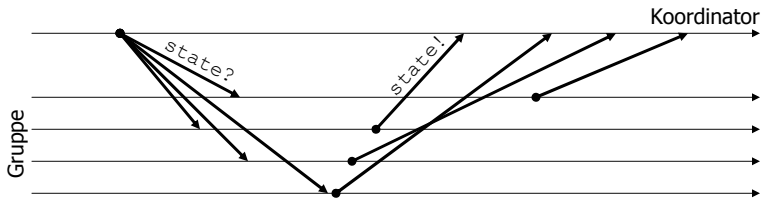
  - Gummibandtransformation?

- Konsistenz eines Schnitts
  - Ein Schnitt S ist konsistent, wenn gilt

$$e \in S \wedge e' <_k e \Rightarrow e' \in S$$

Schnappschuß: Grundidee

- Prozeßgruppe und einen Koordinator
- „Einfärben" von Nachrichten und Prozessen
  - Rot: Links eines konsistenten Schnittes
  - Schwarz: Rechts vom Schnitt
- Koordinator sendet Zustandsabfrage an Gruppe
  - Empfänger wechselt die Farbe
  - Sendet Zustand zurück an Koordinator



Verteilte Systeme, Sommersemester 1999 Folie 9.7

Schnappschuß-Algorithmus

- Annahmen
  - |G| = n
  - Keine Nachrichtenverluste, keine Abstürze
- Koordinator:

```
Schnappschuß () {
    color = initiale Farbe;
    Multicast(state?,color) to G;
    for (i=0;i<n;i++) {
        Receive(state) from p;
        Ergänze verteilten Zustand s;
    }
    color = nächste Farbe;
    return s;
}
```

- Prozeß P:

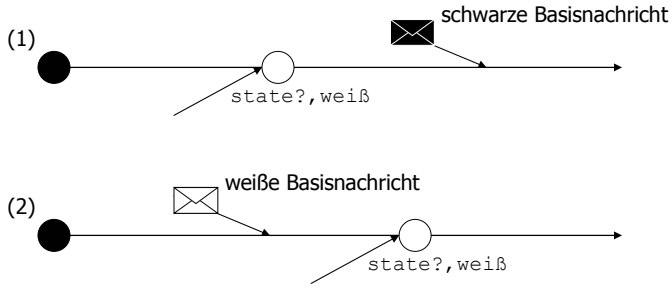
```
Receive(G,state?,color) from Koordinator:
    my_color = color;
    Unicast(state!) to Koordinator;
```

Verteilte Systeme, Sommersemester 1999 Folie 9.8

Fehlt noch etwas?

Einfärben

- Prozeß P ist schwarz: `my_color = schwarz`
- P empfängt `(state?,weiß)` von Koordinator
- Prozesse versenden gefärbte Basisnachrichten
- Szenarien:



Fall 1: Nachträgliche schwarze Nachrichten

- Gehören noch zum übermittelten schwarzen Zustand
  - An Koordinator weiterleiten
  - Koordinator muß Terminierung bzgl. schwarzer Basisnachrichten Abwarten (=nachrichtenbasierte Terminierung)
- Verteilter Zustand
  - Einzelne Zustände der beteiligten Prozesse
  - Im Netz befindlichen Nachrichten

Fall 2: Schwarzer Prozeß empfängt weiße Nachricht

- Zwei Lösungsvarianten
- Weiße Nachricht verzögern, bis `(state?,weiß)` vom Koordinator empfangen wird
  - Prozeß bleibt vorerst schwarz
  - Weiße Nachrichten müssen zwischengespeichert werden
- Weiße Nachricht ist implizit `(state?,weiß)` vom Koordinator
  - Prozeß wird sofort weiß
  - Kann empfangene Nachricht sofort verarbeiten
  - Zustand an Koordinator übermitteln
  - Nachträglich eintreffende schwarze Nachrichten an Koordinator
  - Später eintreffende Nachricht `(state?,weiß)` ignorieren

Bemerkungen und Fragen

- Mehrere Initiatoren parallel
  - Zusammenfassen mehrerer Wellen ggf. möglich
  - Wann?
- Was ist der Zustand?

Election

- Menge von Prozessen muß sich dynamisch für einen Repräsentanten (Gewinner) entscheiden
- Ergebnis
  - Gewinner weiß, daß er gewonnen hat
  - Alle anderen wissen, wer gewonnen hat
- Brechung der Symmetrie
  - Vor- und Nachteile?
- Fairneß wird nicht verlangt
- Election kann von mehreren gleichzeitig gestartet werden
- Verteiltes Minimumproblem (Maximum)
  - „Wert" eines Prozesses: Identifikation, Last, ...

Ein einfacher Election-Algorithmus

- Election bzgl. der Prozeßidentifikation
- Jeder Prozeß besitzt lokale Variable M
  - Initial -∞
  - Enthält am Ende Nummer des Gewinners
- P_k startet Election:

```
M := k; Send(M) to direkte Nachbarn;
```

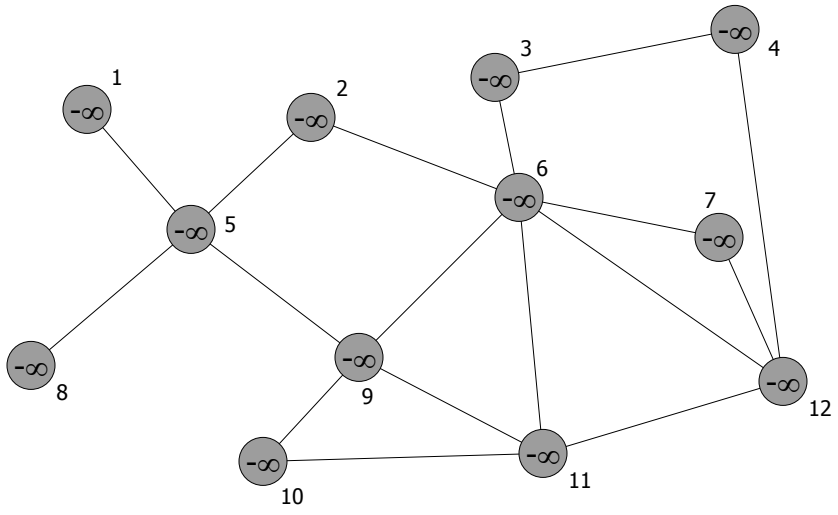
- P_k empfängt Nachricht N:

```
if (M < N) { M := N; Send(M) to direkte Nachbarn; }
```

- Wann ist der Algorithmus zu Ende?

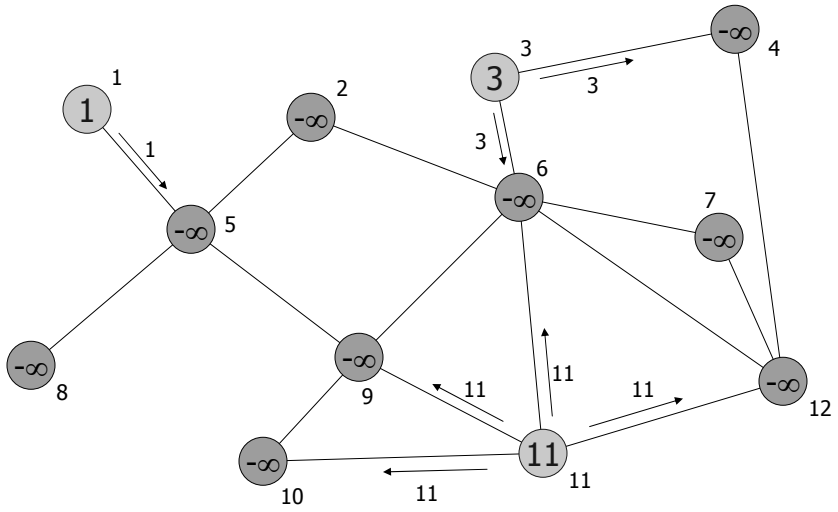
Verteilte Systeme, Sommersemester 1999 Folie 9.15

Beispiel: Ausgangssituation



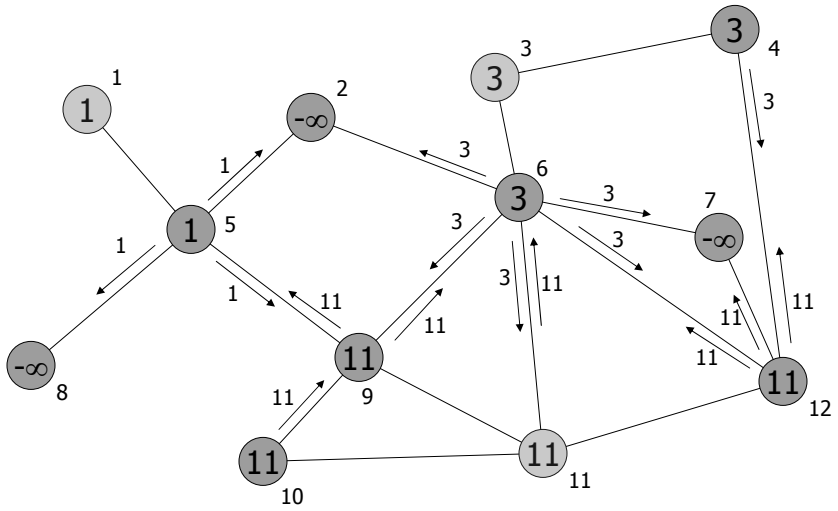
Verteilte Systeme, Sommersemester 1999 Folie 9.16

Beispiel: 1, 3 und 11 beginnen „gleichzeitig" Election



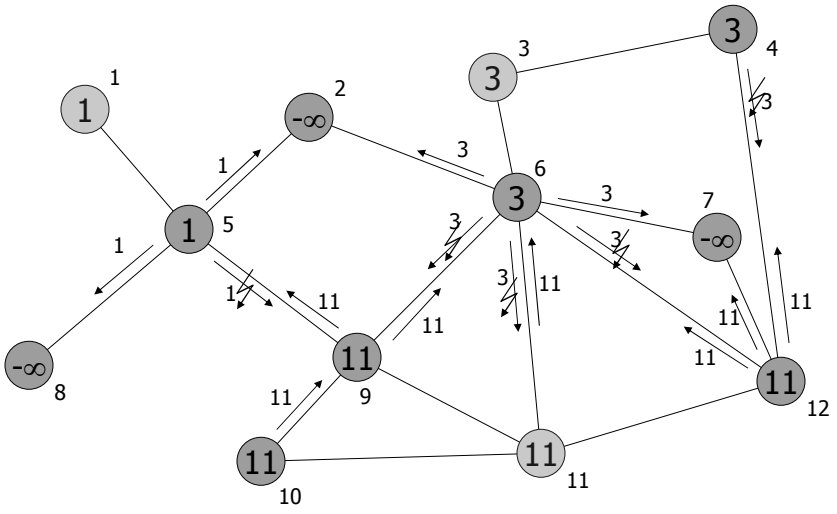
Verteilte Systeme, Sommersemester 1999 Folie 9.17

Beispiel: Welcher Löwe brüllt am lautesten?

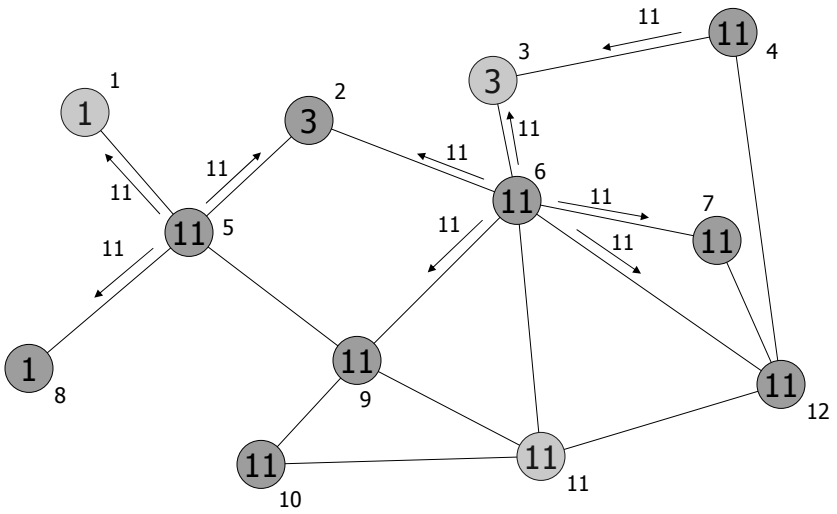


Verteilte Systeme, Sommersemester 1999 Folie 9.18

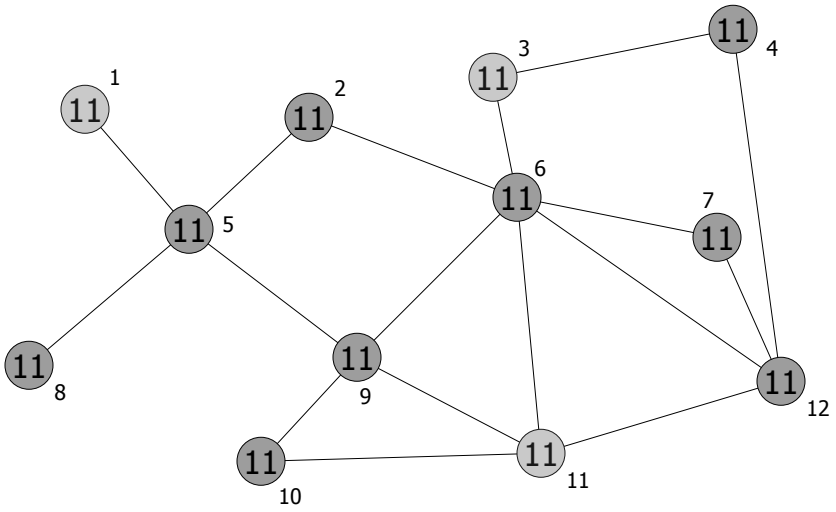
Beispiel: Message Extinction



Beispiel: ... und so weiter

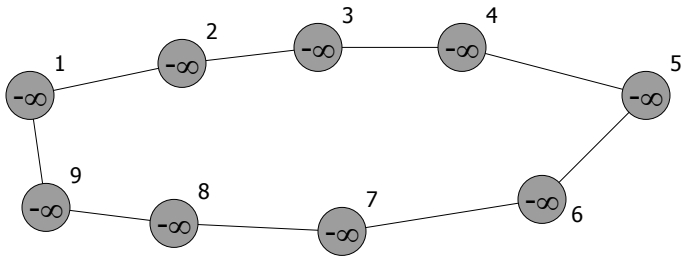


Beispiel: Fertig?



Verteilte Systeme, Sommersemester 1999 Folie 9.21

Election im Ring



- Effiziente Variante des einfachen Election-Algorithmus
- Abschätzung des mittleren Nachrichtenaufkommen aufwendig
- Übung: Alle Prozesse nehmen an der Wahl teil
  - Minimale Nachrichtenanzahl?
  - Maximale Nachrichtenanzahl?

Verteilte Systeme, Sommersemester 1999 Folie 9.22

Weitere Election-Algorithmen

- Reguläre Graphstrukturen
  - Häufig Varianten des einfachen Election-Algorithmus
- Allgemeine Graphstrukturen
  - Grundlage Echo-Algorithmus
  - Verschmelzen von Echo-Wellen bei „gleichzeitigem" Wahlbeginn
- Literatur
  - Friedemann Mattern, *Verteilte Kontrollalgorithmen*, Springer-Verlag

Zusammenfassung

- Lösungen für verteilte Problemstellungen häufig nicht trivial
  - Fehlende globale Zeit
  - Fehlender globaler Zustand
  - Netztopologie
  - Multicast- und Broadcast-Eigenschaften
  - ...
- Forschungsgebiet „Verteile Algorithmen"
  - Ziel Baukastensystem nur bedingt erreicht
    - Anbindung an die Anwendung?
    - Effizienz
    - Fehlende Dynamik
    - ...
- Erlernen der Konzepte verteilter Programmierung