

# 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, Wintersemester 2000/2001

Folie 9.3

## Schnappschußalgorithmen

### Verteilter Zustand

#### Schnappschuß eines globalen Zustands

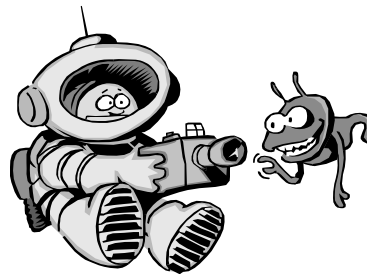
- Lokale Prozeßzustände
- Unterwegs befindliche Nachrichten

### Breites Anwendungsfeld

- Verteilte Terminierung
- Verklemmung
- Wechselseitiger Ausschluß
- ...

### Nachteile

- Meist hohes Nachrichtenaufkommen
- Nicht „Verteilt“ genug



Verteilte Systeme, Wintersemester 2000/2001

Folie 9.4

### Schnitte im Zeitdiagramm

Abfrage erreicht nicht alle Prozess gleichzeitig

Äquivalente Berechnung

– Gummibandtransformation

Verteilte Systeme, Wintersemester 2000/2001 Folie 9.5

### Inkonsistente und konsistente Schnitte

Inkonsistent: Nachrichten aus der Zukunft

– Gummibandtransformation?

Konsistenz eines Schnitts

- Ein Schnitt S ist konsistent, wenn gilt

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

Verteilte Systeme, Wintersemester 2000/2001 Folie 9.6

## Schnappschuß: Grundidee

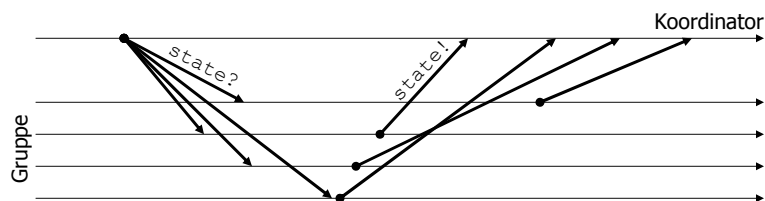
Prozeßgruppe und einen Koordinator

„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, Wintersemester 2000/2001

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, Wintersemester 2000/2001

Folie 9.8

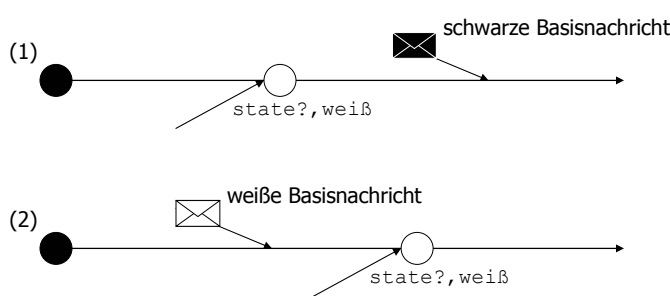
Fehlt noch etwas?



Verteilte Systeme, Wintersemester 2000/2001 Folie 9.9

Einfärben

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



(1) A black process (filled circle) sends a message labeled `state?, weiß` to a white process (open circle). The white process then sends a "schwarze Basisnachricht" (black message icon) to the right.

(2) A black process (filled circle) sends a "weiße Basisnachricht" (white message icon) to a white process (open circle). The white process then sends a message labeled `state?, weiß` to the right.

Verteilte Systeme, Wintersemester 2000/2001 Folie 9.10

## 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

Weißer Nachricht verzögern, bis  $(state?, weiß)$  vom Koordinator empfangen wird

- Prozeß bleibt vorerst schwarz
- Weiße Nachrichten müssen zwischengespeichert werden

Weißer 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

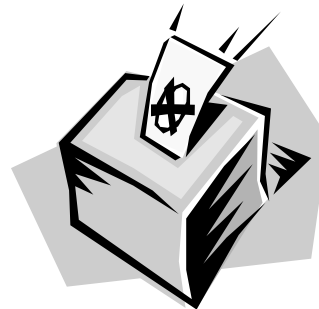
- Vor- und Nachteile?

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  $-\infty$
- Enthält am Ende Nummer des Gewinners

$P_k$  startet Election:

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

$P_k$  empfängt Nachricht N:

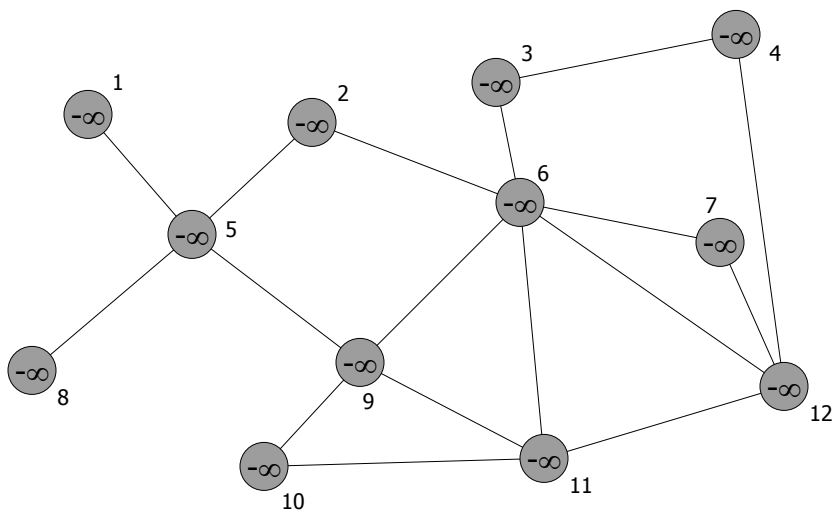
```
if (M < N) { M := N; Send(M) to direkte Nachbarn; }
```

Wann ist der Algorithmus zu Ende?

Verteilte Systeme, Wintersemester 2000/2001

Folie 9.15

## Beispiel: Ausgangssituation

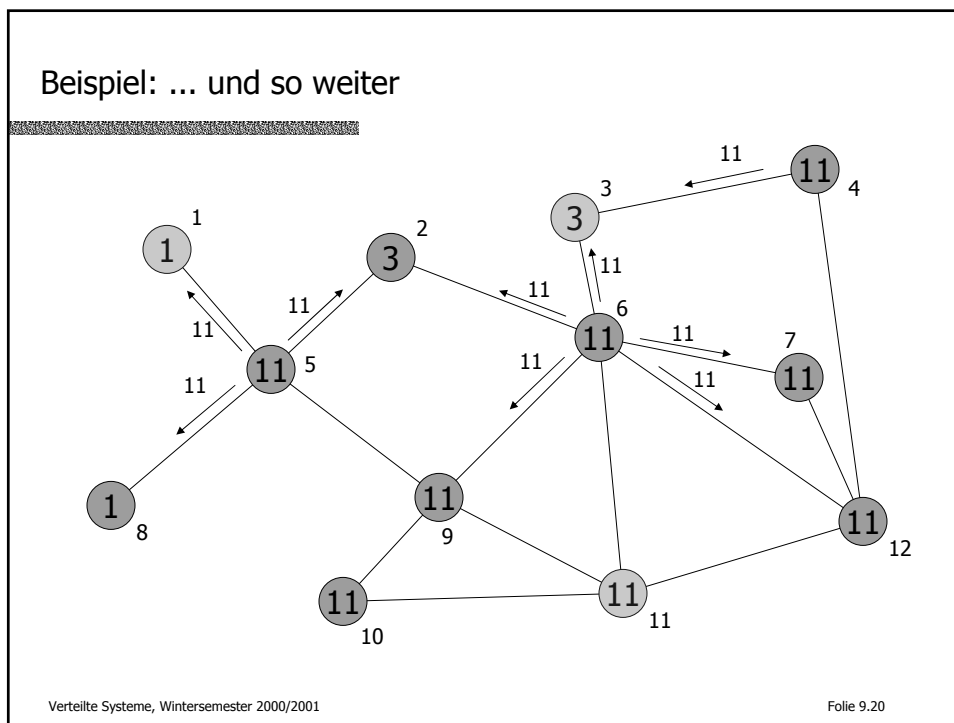
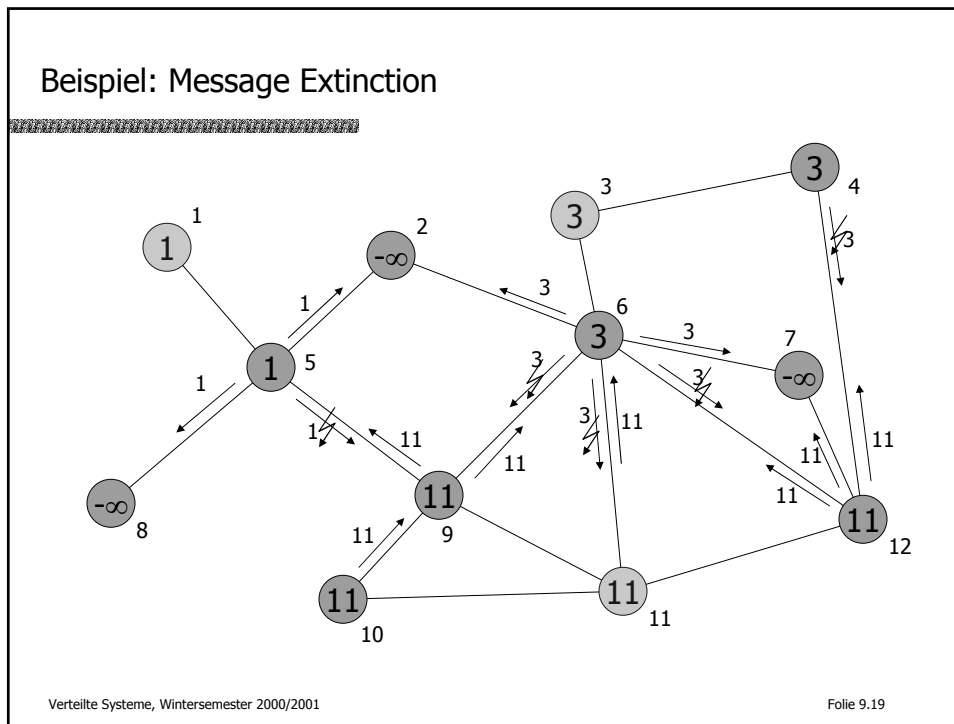


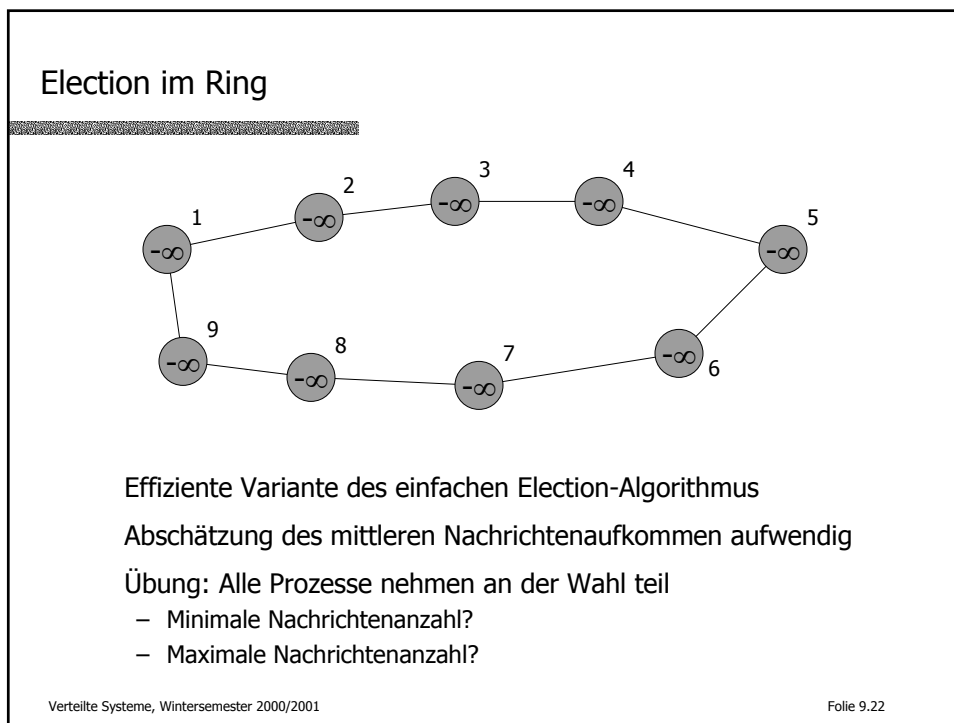
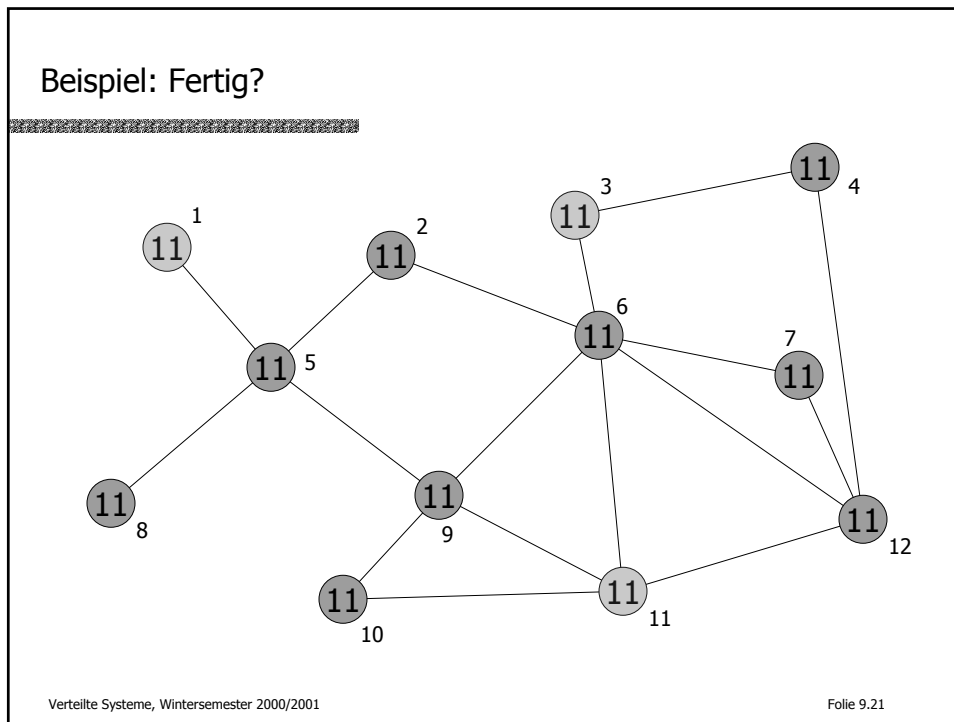
Verteilte Systeme, Wintersemester 2000/2001

Folie 9.16









## 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 „Verteilte Algorithmen“

- Ziel Baukastensystem nur bedingt erreicht
  - Anbindung an die Anwendung?
  - Effizienz
  - Fehlende Dynamik
  - ...

### Erlernen der Konzepte verteilter Programmierung