



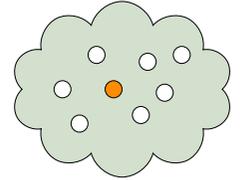
## Verteilte Betriebssysteme

### 9. Kapitel Auswahl

Matthias Werner  
Professur Betriebssysteme

## 9.1 Einführung

- ▶ Viele verteilte Algorithmen und Anwendungen laufen nicht vollständig symmetrisch ab, sondern benötigen einen herausgehobenen Teilnehmer
- ▶ **Beispiele:**
  - ▶ Initiale Festlegung eines Tokenhalters (z.B. ABCAST → Kapitel 5)
  - ▶ Wahl eines Koordinators für zentralisierten Algorithmus (z.B. gegenseitiger Ausschluss → Kapitel 8)
  - ▶ ...
- ▶ Ist dieser besondere Teilnehmer nicht *a priori* bestimmt<sup>1</sup>, muss er zur Laufzeit bestimmt werden
- ▶ Algorithmen, die dies leisten werden **Auswahlalgorithmen** (leader election algorithms) genannt



<sup>1</sup>...was dann einen *single point of failure* darstellt

## Problem

### Auswahl

1. Es soll jeder irgendwann jeder Knoten wissen, ob er ausgewählt ist oder nicht
  2. Es soll nur ein Knoten ausgewählt sein
- ▶ Etwas formaler:
    - ▶ Jeder Knoten ist in einem von drei Zuständen: *unentschieden*, *ausgewählt* oder *nicht ausgewählt*
    - ▶ Jeder Knoten startet im Zustand *unentschieden*
    - ▶ Jeder Knoten geht genau einmal in einen anderen Zustand (*ausgewählt* oder *nicht ausgewählt*) (finaler Zustand)

## Problemvarianten

- ▶ Je nach Variante ist die Lösung des Auswahlproblems **trivial** bis **unmöglich**
- ▶ Z.B.:
  - ▶ Ein Algorithmus mit bekannten Identitäten im vollvermaschten System ist trivial
  - ▶ Im anonymen Ring ist ein deterministischer Algorithmus unmöglich
- ▶ Wir betrachten verschiedene Varianten

## Problemvarianten (Forts.)

- ▶ **Identität:** Haben die Knoten wohlunterschiedene Identitäten?
  - ▶ Ja → **MAX-Algorithmen**
  - ▶ Nein → **anonyme Algorithmen**
- ▶ **Anzahl** der Teilnehmer?
  - ▶ Unbekannt
  - ▶ Bekannt
- ▶ **Determinismus** des Algorithmus
  - ▶ **Deterministische** Algorithmen
  - ▶ **Randomisierte** Algorithmen
- ▶ Sonstiges:
  - ▶ **Synchronisation** (z.B. FIFO-Kanäle, beschränkte Kommunikationszeiten, ...)
  - ▶ **Topologie** (z.B. irregulär, vollvermascht, Ring, Baum, Torus, ...)

## 9.2 MAX-Algorithmen

- ▶ MAX-Algorithmen sind (im Gegensatz zu anonymen Algorithmen) immer einfach
- ▶ Grundidee:

Jeder nennt allen seine *ID*, die größte *ID* gewinnt.

- ▶ Deshalb konzentrieren wir uns auf die **Effektivität**
  - ▶ Anzahl der Nachrichten?
  - ▶ Zeitkomplexität?

## Auswahlalgorithmus für beliebige Topologien

```

1 Ip: {Mp == 0}
2 Mp := p;
3 SEND <Mp> TO all neighbours;
4
5 Rp: {a message <j> arrived}
6 IF Mp < j THEN
7   Mp := j;
8   SEND <Mp> TO all neighbours;
9 FI

1 Tp: {termination was discovered}
2 IF Mp == p THEN
3   "I am the master"
4 ELSE
5   "Mp is the master"
6 FI
    
```

Aber wie?

- ▶ Jeder Prozess hat eine eindeutige Identität *p* und eine lokale Variable *M<sub>p</sub>*, die initial 0 ist
- ▶ *I<sub>p</sub>* wird von den Initiatoren ausgeführt
- ▶ Nachdem *R<sub>p</sub>* ausgeführt wurde, kann *p* nicht mehr Initiator werden ⇒ *p* kann dann auch nicht mehr gewinnen
- ⇒ der höchste Initiator gewinnt

## Auswahlalgorithmus für beliebige Topologien (Forts.)

- ▶ **Problem:** Feststellung der Terminierung
  - ▶ Terminierung ist nichttrivial
  - ▶ Algorithmen zur Terminierungsfeststellung (⇒ späteres Kapitel) können genutzt werden
  - ▶ Auslösung nach „Verdacht“ ⇒ Timeout
- ▶ **Komplexität**
  - ▶ Über jede Kante geht für jeden Initiator höchstens eine Nachricht
  - ▶ Schlechtester Fall: maximale Anzahl von Kanten und Initiatoren
  - ▶ Bei *n* Knoten:  $\mathcal{O}(n^3)$
  - ▶ Aufwand für Terminierungserkennung kommt noch hinzu

## Echo-Auswahlalgorithmus

- ▶ Nutzung des **Echo-Algorithmus** zur Auswahl (→ Kapitel 8)
  - ▶ Funktioniert mit beliebigen, zusammenhängenden Topologien
  - ▶ Jeder Initiator startet eine Instanz des Echo-Algorithmus
  - ▶ Explorer und Echos führen die **Identität** des Initiators mit
  - ▶ Schwächere Nachrichten (Explorer und Echos) werden **nicht** weitergegeben sondern verschluckt ⇒ **Nachrichtenauslöschung**

## Echo-Auswahlalgorithmus (Forts.)

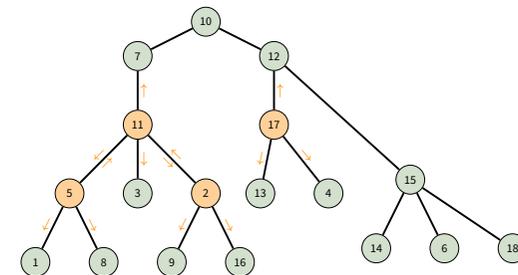
- ▶ Stärkste Welle setzt sich durch und **terminiert beim Gewinner**
  - ▶ Gewinner weiß, dass er gewonnen hat
  - ▶ Empfängt ein Initiator einen stärkeren Explorer, dann weiß er, dass er verloren hat
- ▶ Wie erfahren die **Verlierer** von der Terminierung?
  - ▶ Alle anderen Wellen stagnieren irgendwo endgültig
  - ▶ Zumindest der stärkste Initiator sendet kein Echo für die schwächeren Wellen
  - ▶ Bei **keinem** anderen Initiator terminiert sein Echo-Algorithmus
  - ▶ Wenn dies nicht ausreicht, muss der Gewinner die anderen in einer **weiteren Welle** informieren
- ▶ **Komplexität** ist im schlechtesten Fall wieder  $\mathcal{O}(n^3)$  Nachrichten (besser, wenn bereits Spannbaum besteht)

## Modifizierter Echo-Algorithmus bei Baum-Topologie

- ▶ Nutzen Baum-Topologie für Wellen-Algorithmus
- ▶ Drei Phasen
  - ▶ **Explosionsphase**  
Wahlaufforderung wird zu den Blättern propagiert
  - ▶ **Kontraktionsphase**  
Von den Blättern wird das Maximum der bisher eingesammelten Identitäten zum Zentrum propagiert
  - ▶ **Informationsphase**  
Verbreitung des echten Maximums vom Zentrum aus an alle Knoten im Netzwerk

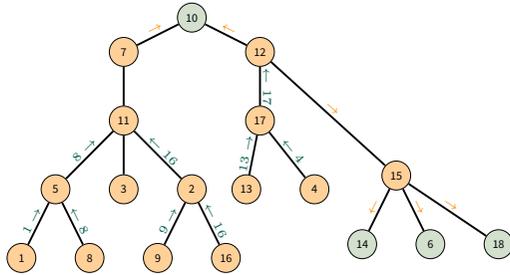
## Explosionsphase

- ▶ Die Explosion startet bei mehreren Knoten (Initiatoren)
- ▶ Beim erstmaligen Erhalt einer Explosionsnachricht wird diese an alle anderen Nachbarn weitergegeben
- ▶ Die Explosionswellen vereinigen sich dort, wo sich Explosionsnachrichten auf einer Kante begegnen



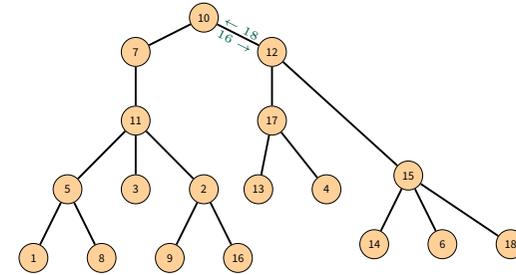
## Kontraktionsphase

- ▶ Blätter antworten auf eine Explosionsnachricht sofort mit ihrer eigenen Identität
- ▶ Alle anderen Knoten senden über die letzte verbliebene Kante das Maximum aus den über die anderen Kanten erhaltenen Identitäten und ihrer eigenen Identität



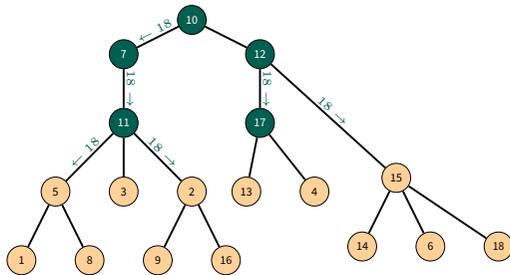
## Ende der Kontraktionsphase

- ▶ Auf **genau einer** Kante begegnen sich zwei Kontraktionsnachrichten mit **unterschiedlichen** Maxima
- ▶ Beide empfangenden Knoten kennen danach das echte Maximum



## Informationsphase

- ▶ Von beiden Knoten aus, wird das Maximum ins Netzwerk geflutet, wobei die Kante zwischen ihnen ausgespart wird



## Nachrichtenkomplexität

- ▶ Explosionsphase:  $(n - 1) + (k - 1) = n - 2 + k$ 
  - ▶ Eine Nachricht über jede Kante
  - ▶ Ausnahmen: 2 Nachrichten über die  $k - 1$  Begegnungskanten
- ▶ Kontraktionsphase:  $(n - 1) + 1 = n$ 
  - ▶ Eine Nachricht über jede Kante
  - ▶ Ausnahme: Zwei Nachrichten über die Zentrums-kante
- ▶ Informationsphase:  $(n - 1) - 1 = n - 2$ 
  - ▶ Eine Nachricht über jede Kante
  - ▶ Ausnahme: Keine Nachricht über die Zentrums-kante
- ▶ Insgesamt  $3n + k - 4$  Nachrichten ( $\Rightarrow \mathcal{O}(n)$ )

## 9.3 Auswahl in anonymen Netze

- ▶ In anonymen Netzen haben Knoten **keine** dauerhaften **eindeutigen** Identitäten
- ▶ Lässt sich dann noch ein eindeutiger Gewinner der Auswahl bestimmen?
  - ▶ Mit deterministischen Algorithmen: nein  
(solange nicht eine bekannte Topologie eine indirekte Unterscheidung gibt)
  - ▶ Mit randomisierten Algorithmen: vielleicht

## Unterscheidung randomisierter Algorithmen

- ▶ **Las-Vegas-Algorithmen**
  - ▶ Liefern **immer** ein korrektes Ergebnis
  - ▶ Gegebenenfalls ist aber die worst-case Zeitkomplexität **unbeschränkt** ⇒ Abschwächung der **Terminierungsforderung**
- ▶ **Monte-Carlo-Algorithmen**
  - ▶ Dürfen mit beschränkter Wahrscheinlichkeit ein **inkorrektes** Ergebnis liefern
  - ▶ Worst-case Laufzeit ist beschränkt
- ▶ Wir betrachten hier Las-Vegas-Algorithmen für das Auswahlproblem

## Unmöglichkeit anonymer deterministischer Algorithmen

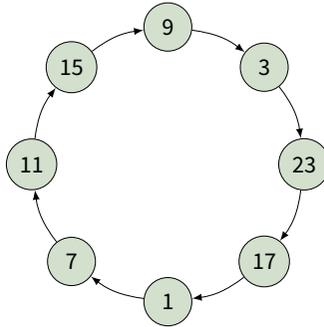
- ▶ Betrachten vollständig symmetrische Topologie:
  - ▶ anonymer Ring (Ringgröße darf bekannt sein)
  - ▶ synchrone Kommunikation
- ▶ **Annahme:** Es gibt einen **deterministischen** Algorithmus  $\mathcal{A}$ , der das Auswahlproblem löst
- ▶ **Beweis durch Widerspruch:**
  - ▶ Da alle Teilnehmer identisch sind, starten alle Teilnehmer von  $\mathcal{A}$  im gleichen Zustand
  - ▶ In jeder Runde von  $\mathcal{A}$  führen alle Teilnehmer die gleichen Schritte aus ⇒ sind im gleichen Zustand
  - ▶ Wenn jemand sich zum Sieger erklärt, erklären sich alle zum Sieger ⇒ Widerspruch
- ▶ **Lösung:** probabilistische Algorithmen

## Las Vegas-Auswahl für anonyme Ringe

- ▶ Es gibt aber selbst für eine symmetrische Topologie wie einen Ring Algorithmen
- ▶ Betrachten Algorithmus von ITAI und RODEH ⇒ basiert auf MAX-Algorithmus von CHANG und ROBERTS
- ▶ Betrachten deshalb zuerst CHANG und ROBERTS:
  - ▶ Topologie: Ring mit Größe  $n$
  - ▶ Jeder Knoten hat eindeutige UID
  - ▶ Kommunikation erfolgt stets in nur einer Richtung, z.B. im Uhrzeigersinn

## Exkurs: Chang/Roberts

- ▶ Jeder Knoten ist in einem der drei Zustände:  $Z \in \{\text{Nichtteilnehmer, Teilnehmer, Sieger}\}$ 
  - ▶ Am Anfang ist jeder Knoten **Nichtteilnehmer**

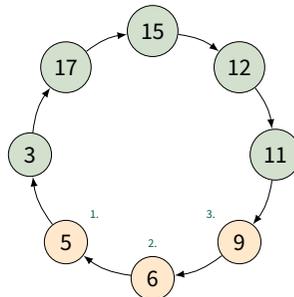


## Exkurs: Chang/Roberts – Algorithmus

- ▶ Ein (oder mehrere) Knoten bemerk(t/en) das Fehlen eines Koordinators (Timeout), erklär(t/en) sich zum **Teilnehmer**, und verschick(t/en) eine Wahlnachricht mit seiner/ihrer UID
- ▶ Beim Empfang einer Wahlnachricht reagiert ein Knoten wie folgt:
  - ▶ Ist die UID in der Nachricht größer als die eigene, wird man **Teilnehmer** (falls man es noch nicht ist) und die Nachricht **weitergeleitet**
  - ▶ Ist die UID in der Nachricht kleiner als die eigene und man ist noch nicht **Teilnehmer**, wird man **Teilnehmer** und die UID in der Nachricht durch die eigene **UID ersetzt** und die Nachricht **weitergeleitet**
  - ▶ Ist die UID in der Nachricht kleiner als die eigene und man ist bereits **Teilnehmer**, wird die Nachricht **ignoriert**
  - ▶ Ist die UID in der Nachricht gleich der eigenen, **erklärt** man sich zum **Sieger**
- ▶ Ein **Sieger** sendet eine Sieger-Nachricht mit seiner UID, die von allen weitergeleitet wird
  - ▶ Wenn der **Sieger** seine eigene Nachricht empfängt, terminiert der Algorithmus

## Exkurs: Chang/Roberts – Worst-case Nachrichtenkomplexität

- ▶ **Annahme:**  $k$  Initiatoren
- ▶ Worst case tritt ein, wenn die Initiatoren in absteigender Reihenfolge auf dem Ring angeordnet sind und in aufsteigender Reihenfolge die Auswahl initiieren
  - ▶  $k$ -größter Initiator
    - ➔  $n - (k - 1)$  Nachrichten
  - ▶ ...
  - ▶ 3.-größter Initiator
    - ➔  $n - 2$  Nachrichten
  - ▶ 2.-größter Initiator
    - ➔  $n - 1$  Nachrichten
  - ▶ größter Initiator
    - ➔  $n$  Nachrichten



## Exkurs: Chang/Roberts – Worst-case Nachrichtenkomplexität (Forts.)

- ▶ Nachrichtenkomplexität bei  $k$  Initiatoren

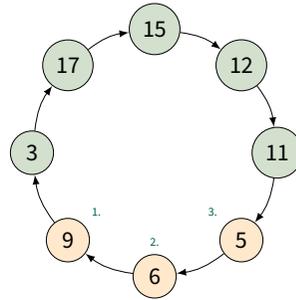
$$\begin{aligned}
 N &= n + (n - 1) + (n - 2) + \dots + (n - (k - 1)) \\
 &= \frac{n(n - 1)}{2} - \frac{(n - k)(n - k + 1)}{2} \\
 &= \frac{2nk - k^2 + k}{2} \\
 &= nk - \frac{k(k - 1)}{2}
 \end{aligned}$$

- ➔  $\mathcal{O}(n^2)$  bei Ringgröße  $n$  und  $k = n$
- ▶ Noch  $n$  zusätzliche Nachrichten für die Gewinnbenachrichtigung

$$\sum_{i=1}^n i = \frac{n(n + 1)}{2}$$

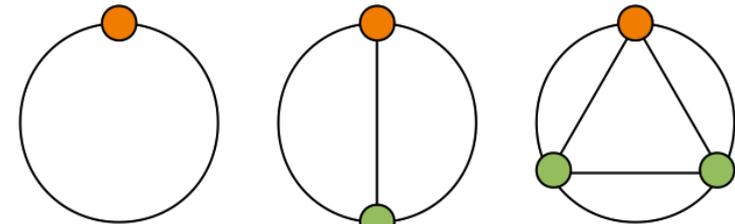
## Exkurs: Chang/Roberts – Best-Case Nachrichtenkomplexität

- ▶ Best case tritt ein, wenn die Initiatoren in aufsteigender Reihenfolge auf dem Ring angeordnet
- ▶ Alle außer dem größten Initiator verursachen nur eine Nachricht
- ▶ Der größte Initiator verursacht  $n$  Nachrichten  $\Rightarrow n + k - 1$  Nachrichten für die eigentliche Auswahl
- ▶ Wieder noch  $n$  zusätzliche Nachrichten für die Gewinnbenachrichtigung



## Exkurs: Chang/Roberts: Mittlere Komplexität

- ▶ Nachrichtenkomplexität über alle möglichen  $n!$  Permutationen der Nummern auf dem Ring
- ▶ Informale Argumentation
- ▶ Größter Initiator  $\Rightarrow n$  Nachrichten (immer)
- ▶ 2.-größter Initiator  $\Rightarrow \frac{n}{2}$  Nachrichten im Mittel
- ▶ 3.-größter Initiator  $\Rightarrow \frac{n}{3}$  Nachrichten im Mittel
- ▶ ...
- $\Rightarrow i$ -größter Initiator  $\Rightarrow \frac{n}{i}$  Nachrichten im Mittel



● betrachteter Initiator    ● größerer Initiator

## Exkurs: Chang/Roberts: Mittlere Komplexität (Forts.)

- ▶ Die average-case Nachrichtenkomplexität beträgt daher

$$nH_k \approx n \ln k$$

mit  $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k}$

- ▶ Für unidirektionale Ringe ist dies optimal
- ▶  $H_k$  ist die  $k$ -te **Harmonische Zahl** ( $\Rightarrow$  **Harmonische Reihe**)
- ▶ Für sehr große Ringe werden fast nie mehr Nachrichten als im Durchschnitt benötigt (Rotem et al.)
- ▶ Wieder:  $n$  zusätzliche Nachrichten für die Gewinnbenachrichtigung

## ITAI und RODEH

Zurück zu ITAI und RODEH

- ▶ **Annahmen:**
  - ▶ Anonymer Ring: kein Knoten hat eine (permanente) Identität
  - ▶ Ringgröße  $n$  muss bekannt sein
  - ▶ Jeder Knoten besitzt einen Zufallsgenerator  $\Rightarrow$  Randomisierung
- ▶ **Unterschiede zu CHANG/ROBERTS:**
  - ▶ Verfahren wird in mehreren Runden durchgeführt, von denen jede etwa einer Wahl in CHANG/ROBERTS entspricht
  - ▶ Nachricht enthält neben einer ID noch einen Hopcount  $h$  und ein Flag  $f$

## ITAI und RODEH (Forts.)

- ▶ (Jeder) Initiator wählt eine **zufällige** ID und schickt Nachricht  $\langle ID, f = 1, h = 0 \rangle$
- ▶ Jeder Knoten inkrementiert beim Weiterleiten einer Nachricht den Hopcount  $h \leftarrow h + 1$
- ▶ Empfängt ein Knoten eine Nachricht mit seiner eigenen Identität, so werden zwei Fälle unterschieden
  - ▶ Ist  $h \neq n$ , so gibt es mindestens einen anderen Knoten mit gleicher Identität
    - ▶  $f$  wird auf 0 gesetzt und die Nachricht weitergeleitet
  - ▶ Ist  $h = n$ , so hat er die Wahl gewonnen
    - ▶ Ist  $f = 1$ , dann ist er der einzige Gewinner und kann die Sieger-Nachricht losschicken
    - ▶ Ist  $f = 0$ , dann gibt es mehrere Gewinner

## ITAI und RODEH (Forts.)

- ▶ Es werden iterativ Wahlgänge durchgeführt, bis ein eindeutiger Sieger feststeht
  - ▶ Nur die **Sieger** des aktuellen Wahlgangs nehmen aktiv am nächsten Wahlgang teil
    - ➔ Hierfür wählt jeder Sieger eine **neue**, zufällige Identität und sendet diese um den Ring
  - ▶ Ausgeschiedene Knoten sind **passiv** und leiten lediglich Nachrichten weiter (mit inkrementiertem Hop-Count)
  - ▶ Nachrichten aus kleineren Wahlgängen werden **verschluckt**

## ITAI und RODEH – Diskussion

- ▶ Es gibt keine Garantie, dass der Algorithmus jemals terminiert
- ▶ Aber: Die **Wahrscheinlichkeit**, dass er **nicht** terminiert, geht mit der Zeit gegen 0
- ▶ Grenzwert<sup>2</sup> für den Erwartungswert  $E$  für die Anzahl der Wahlgänge bei Anzahl von möglichen Identitäten  $c = n$

$$E \leq e \frac{n}{n-1}$$

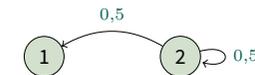
( $e$  ist hier die **Eulersche Zahl**)

<sup>2</sup>Das ist eine Grobabschätzung, es gibt einen Grenzwert  $E \leq 2,441715 \cdot n$

## ITAI und RODEH – Diskussion (Forts.)

Ableitung des Erwartungswerts für  $n = c = 2$

- ▶ Anwendung einer Markow-Kette



$$\begin{aligned}
 E &= \sum_{i=1}^{\infty} i \cdot \frac{1}{2^i} \\
 &= 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2^2} + 3 \cdot \frac{1}{2^3} + \dots \\
 &= 2
 \end{aligned}$$

- ▶ Ableitung ist auch für allgemeines  $n$  möglich

## ITAI und RODEH – Diskussion (Forts.)

- ▶ Kenntnis der Ringgröße  $n$  ist eine sehr starke Anforderung
- ▶ **Abschwächung:** Ringgröße ist nur *ungefähr* bekannt:  $\exists k, k \leq n < 2k$ 
  - ▶ Nachricht mit nichtkleinerer ID wird stets weitergeleitet, bis  $h = 2k - 1$ 
    - ▶ mindestens eine Runde ist garantiert
  - ▶ Das Flag  $f$  wird gelöscht, wenn eine Nachricht mit der eigenen ID und einem Hop-Count von  $h < k$  empfangen wird
    - ▶ Gibt es mehr als einen Sieger, erfährt **mindestens ein** Knoten davon und teilt es den anderen mit

## Algorithmus für beliebige Graphen (Forts.)

- ▶ Jeder Knoten ist immer entweder die **Wurzel** (*root*) eines Baumes, oder hat einen **Vaterknoten**
  - ▶ Vom Sohn auf den Vater ist eine Kante im Baum gerichtet
- ▶ Jeder Knoten hat folgende Variablen, die von den Nachbarn gelesen werden können:
  - $tid$ : Eine ID des Baums, zu dem der Knoten gehört
  - $d$ : Entfernung des Knotens zur Wurzel (für Wurzel:  $d = 0$ )
  - $c$ : Farbe (ein Wert aus  $[0, \dots, 7]$ )
- ▶ Jeder Knoten führt in einer Endlosschleife folgende Schritte aus
  - ▶ Wenn ein Baum mit größerer  $tid$  gefunden wird, schließe dich ihm an  $\Rightarrow$  Teilalgorithmus **Baumfusion**
  - ▶ Wenn ein Fehler erkannt wird, erkläre dich zur neuen Wurzel  $\Rightarrow$  Teilalgorithmus **Fehlererkennung**
  - ▶ Wenn andere Bäume mit gleicher  $tid$  entdeckt werden, vergrößere zufällig die  $tid$  deines Baums  $\Rightarrow$  Teilalgorithmus **Konkurrenzerkennung**

## Algorithmus für beliebige Graphen

- ▶ Der folgende Algorithmus geht auf DOLEV et al [DIM97] zurück
- ▶ Er erlaubt die Lösung des Auswahlproblems für beliebige (zusammenhängende) Topologien
- ▶ Der Algorithmus ist dynamisch und **selbststabilisierend**:
  - ▶ Er **terminiert nie** und findet nach Fehlern (Ausfällen von Knoten oder Kanten) wieder (irgendwann) in einen korrekten Zustand zurück
- ▶ **Idee:** Konstruktion eines gerichteten Baums (*in-tree*), bei dem die Wurzel der Ausgewählte Knoten ist
- ▶ **Annahmen:**
  - ▶ Ein Knoten kann seine Nachbarn über die Kommunikationskanäle lokal identifizieren
  - ▶ Ein Knoten kann Variablen seiner Nachbarn abfragen (RPC); falls nicht, wird der Kanal als fehlerhaft (nicht vorhanden) angesehen

## Algorithmus für beliebige Graphen (Forts.)

- ▶ **Baumfusion/Fehlererkennung**
  - ▶ Frage alle Nachbarn  $i$  ab und ermittle  $tid_{max} = \max_i(tid_i)$  sowie  $d_{min} = \min_{tid_i=tid_{max}}(d_i)$
  - ▶ Ist  $tid_{max}$  kleiner als eigene  $tid$  oder ist sie gleich, aber  $d_{min}$  ist größer als die eigene Distanz  $d$ , erkläre dich zur eigenen Wurzel
  - ▶ Sonst ist der Knoten mit  $tid_i = tid_{max}$  der (ggf. neue) Vater, von dem die  $tid$  geerbt wird; die eigene Distanz wird auf  $d = d_i + 1$  gesetzt
- ▶ **Anmerkungen:**
  - ▶ Falls man selbst vorher eine Wurzel war, wird die Anzahl der Bäume reduziert
  - ▶ Durch die Distanzüberprüfung werden Schleifen oder verschwindende Kanten entdeckt
  - ▶ Nach hinreichend vielen Durchläufen ohne Fehler entsteht ein Wald mit Bäumen mit jeweils gleicher  $tid$

## Algorithmus für beliebige Graphen (Forts.)

### ► Konkurrenzerkennung

- Eine Wurzel wählt eine zufällige Farbe aus und initiiert ein Umfärben des Baums (Wellenverfahren)
- Sobald alle Elemente des Baums das Umfärben quittiert haben, initiiert die Wurzel das nächste Umfärben
- Entdeckt ein Knoten einen Nachbarknoten mit gleicher *tid* aber einer Farbe, die weder die aktuelle noch die vorherige Baumfarbe ist, wird der Wurzel eine Konkurrenz gemeldet
- In diesem Fall vergrößert die Wurzel die *tid* zufällig und propagiert diese neue Baum-ID in ihrem Baum

### ► Anmerkungen

- Da das nächste Umfärben erst beginnt, wenn das letzte beendet ist, zeigt eine andere Farbe als die aktuelle oder letzte einen fremden Baum an
- Da die Farbe jeweils zufällig gewählt wird, sinkt die Wahrscheinlichkeit, dass zufällig die gleichen Farben gewählt werden
- Auch der Konkurrenzbaum kann eine Vergrößerung seiner *tid* initiiert haben; da diese aber um einen zufälligen Wert erfolgt, sinkt die Wahrscheinlichkeit auf gleiche IDs

## Diskussion

- Der Algorithmus ist hier nur informal beschrieben, für formale Betrachtungen und Korrektheitsbeweis siehe [DIM97]
- Hat der finale Baum eine Tiefe von  $D$  und jeder der  $n$  Knoten maximal  $N$  Nachbarn, so ist im fehlerfreien Fall die Rundenzahl bis zu Wahl genau einer Wurzel in  $\mathcal{O}(N \cdot D \cdot \log n)$
- Die Terminierung kann nicht entdeckt werden, wenn nicht vorher die Gesamtanzahl von Knoten bekannt ist

## Andere Ansätze zum Brechen der Symmetrie

- Ist die Topologie nicht symmetrisch und bekannt, so kann dies zum Brechen der Symmetrie unter anonymen Knoten genutzt werden
  - Beispiel: in einem Gitter (Grid) kann nach einem bestimmten Eckknoten gesucht werden
- In der Praxis kommen anonyme Systeme häufig bei Netzen aus Sensorknoten vor
- In diesem Fall können Sensorinformationen zum Symmetriebruch genutzt werden
  - GPS-Koordinaten
  - Relative Position anderer Knoten

## Literatur

-  [Mat89] Mattern, F.: *Verteilte Basisalgorithmen*. Springer, 1989, Kapitel 2
-  [ChRo79] Chang, E. und R. Roberts: „An improved algorithm for decentralized extrema-finding in circular configurations of processes“. *Communications of the ACM*, 22(5)1979, 281–283
-  [DIM97] Dolev, S., A. Israeli und S. Moran: „Uniform Dynamic Self-Stabilizing Leader Election“. *IEEE Transactions on Parallel and Distributed Systems*, 8(4)1997, 424–440