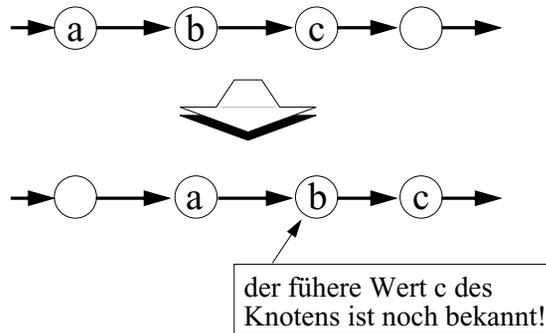


# Variante für unidirektionale Ringe

- Man glaubte zunächst, dass ein Election-Algorithmus auf *unidirektionalen* Ringen mindestens  $O(n^2)$  Nachrichten im Worst-case-Fall benötigt
- Das stimmt nicht: Der Peterson-Algorithmus lässt sich auf unidirektionalen Ringen *simulieren!*

1. "Shift" in Ringrichtung um eine Position bzgl. aktiver Knoten:



2. Nun kann (der neue) Knoten a an seinen Nachbarn b seinen Wert senden - damit kennt b sowohl a als auch c (als hätte b Nachrichten von a und c erhalten!)

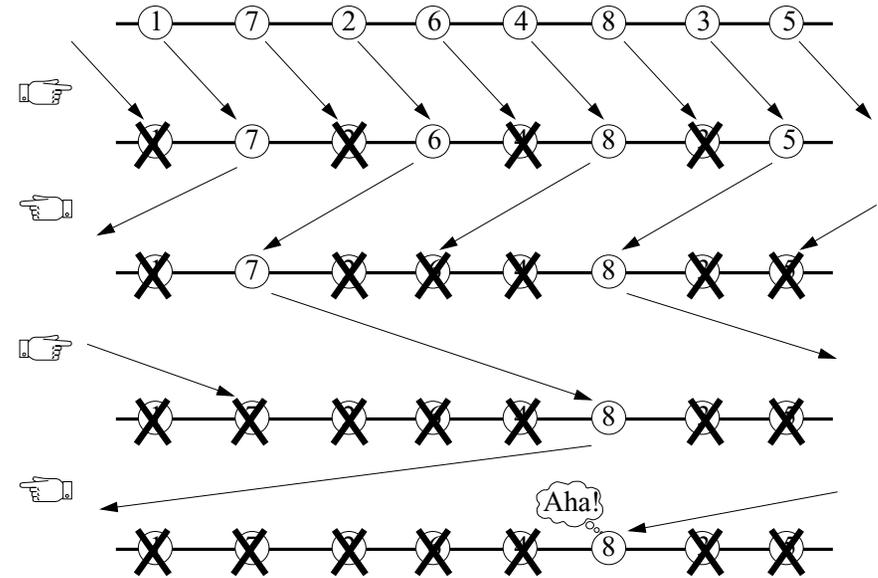
Damit kostet eine solche Phase global auch nur  $2n$  Nachrichten!

# Peterson's Election-Algorithmus (2. Variante)

- Idee einer Optimierung:

- anstatt sich mit beiden Nachbarn "gleichzeitig" zu vergleichen, sollte ein Knoten sich nur dann mit seinem anderen Nachbarn vergleichen, wenn er den ersten Vergleich gewonnen hat

- Phasen im / gegen den Uhrzeigersinn wechseln sich ab:



- lässt sich auch wieder unidirektional simulieren!
- in jeder Phase werden  $n$  Nachrichten gesendet (passive Knoten: "relay")

- Denkbungen:

- 1) Wie kann man auch bei asynchronen Nachrichten und nicht gleichzeitigem Start der Knoten "eine Art" global getakteter Phasen erreichen?
- 2) Man formuliere den Algorithmus aus "Sicht eines Knotens":  
Wie reagiert ein Knoten auf das Eintreffen einer bestimmten Nachricht?
- 3) Man mache sich Gedanken zur Abschätzung der worst-case und der average-case Nachrichtenkomplexität!

# Nachrichtenkomplexität

$$\text{Basis } \phi = (1 + \sqrt{5})/2$$

## - Behauptung:

Für die Anzahl der Phasen  $r$  gilt:  $r \leq \log_{\phi} n + O(1)$

$\implies$  Anzahl der Nachrichten  $\leq 1.44 n \log_2 n + c$

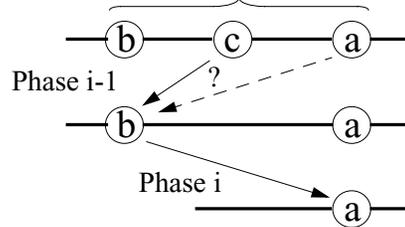
## - Lemma: $a_i \leq a_{i-2} - a_{i-1}$ (für $i > 1$ )

Def: Anzahl Überlebende von Phase  $i$

Anzahl der "Opfer" von Phase  $i-1$

**Bew.:** Betrachte zwei benachbarte Knoten  $a, b$  in Phase  $i$

Gab es einen Knoten zwischen  $a$  und  $b$  in Phase  $i-1$ ?



(1)  $a$  überlebe Phase  $i$

$\implies a > b$

(2)  $b$  hat Phase  $i-1$  überlebt

$\implies b > c$

(1) und (2)  $\implies a \neq c$

Also muss es ein  $c$  geben, das in Phase  $i-1$  Opfer wurde

Hier:  $a$  hat seinen linken Nachbarn in der vorherigen Phase verloren

$\implies$  Für jeden Überlebenden in Phase  $i$  (hier:  $a$ ) gibt es mindestens ein Opfer (hier:  $c$ ) in Phase  $i-1$   $\square$

- Aus  $a_i \leq a_{i-2} - a_{i-1}$  folgt  $a_{i-2} \geq a_{i-1} + a_i$ , also  $a_i \geq a_{i+1} + a_{i+2}$

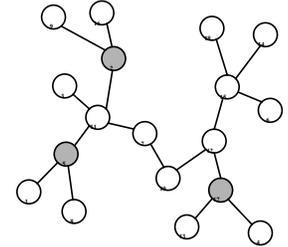
- Ferner gilt  $a_{r-1} = 1 = \text{Fib}(2)$   
 $a_{r-2} \geq 2 = \text{Fib}(3)$  }  $a_{r-3} \geq a_{r-2} + a_{r-1} \geq \text{Fib}(2) + \text{Fib}(3) = \text{Fib}(4)$

- Also:  $n = a_0 \geq \text{Fib}(r+1)$

--> Ungleichung nach  $r$  auflösen!  
 Weil Fib exponentiell zur Basis  $\phi$  wächst  
 $(\text{Fib}(k) \approx \phi^k / \sqrt{5})$ , folgt die Behauptung

# Election auf Bäumen

- Geht dies besser / effizienter als z.B. mit dem Message-extinction-Prinzip für allg. Graphen?
- Und im Vergleich zu den Verfahren auf Ringen?



## - Explosionsphase:

- Election-Ankündigung wird zu den Blättern propagiert

## - Kontraktionsphase

- von aussen zum "Zentrum" das Maximum propagieren

## - Informationsphase (notw.?)

- Zentrum informiert alle Knoten über Gewinner

flooding!

0 für unbeteiligte Knoten

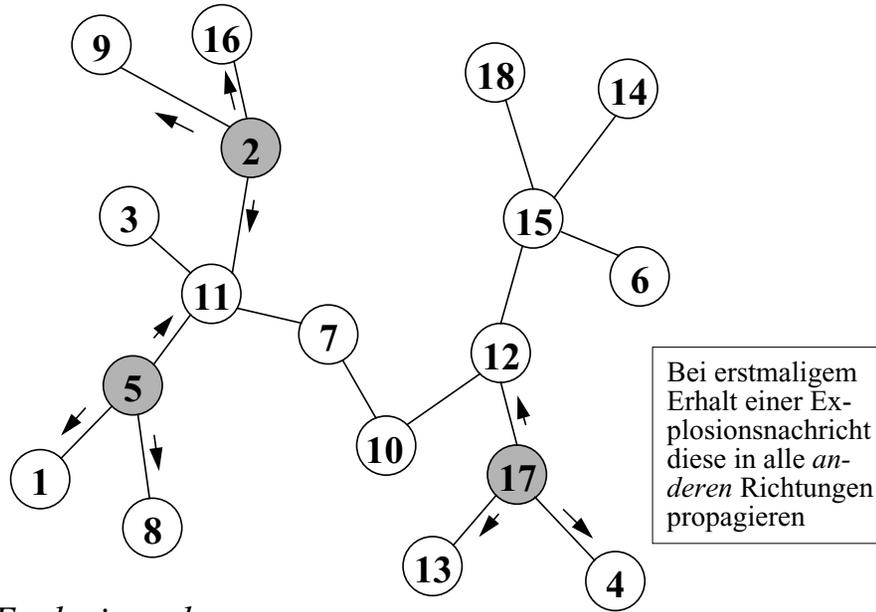
flooding!

- Explosionsphase kann an mehreren Stellen "zünden"

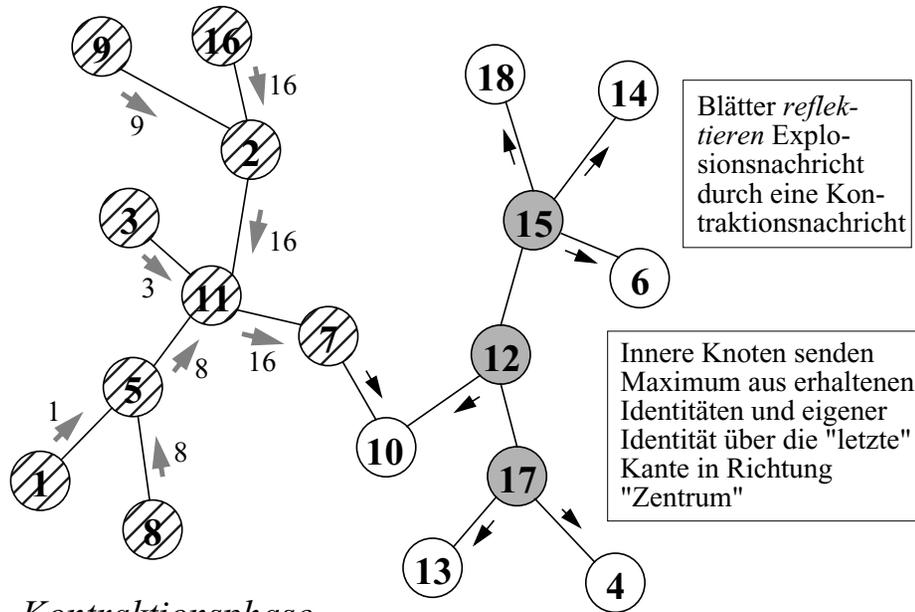
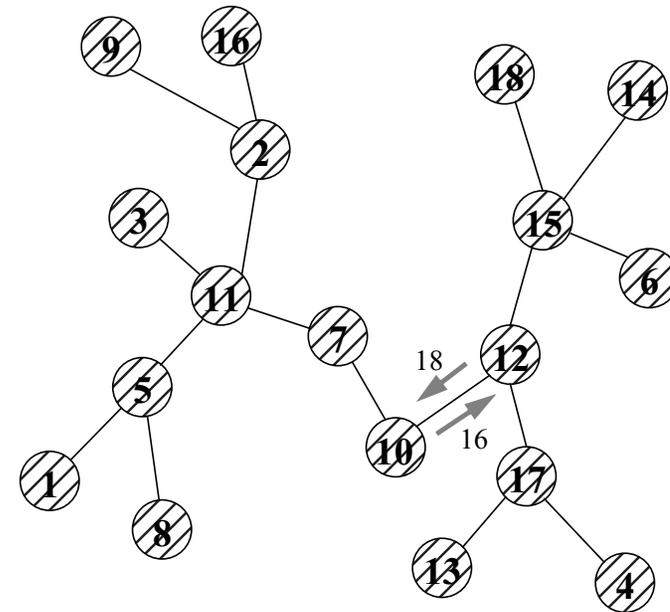
- "Vereinigung" der Explosionsphasen

- Ggf. Teile in Explosionsphase während andere Teile schon in Kontraktionsphase

- Zentrum ist nicht determiniert



Explosionsphase



Kontraktionsphase

- Begegnung zweier Kontraktionsnachrichten auf (genau) einer Kante im Zentrum
- Die beiden Knoten wissen, dass sie nun das Maximum kennen
- Sie können dies nun ggf. per flooding verbreiten (Informationsphase)
- Terminierung der flooding-Phase (falls notwendig) einfach durch erneute Reflexion / Kontraktion ("indirektes acknowledge")

# Nachrichtenkomplexität von Baumelection

Folgender Satz / Beweis ist *falsch* - wieso?

*Beh.:* Der Baumelection-Algorithmus hat bei *einem* Initiator und  $n$  Knoten die Komplexität  $m(n) = 2n - 2$  (ohne Berücksichtigung der Informationsphase)

*Beweis* induktiv:

- 1)  $n=1 \rightarrow m(1) = 0$  ✓ (offensichtlich korrekt)
- 2) Schritt von  $n$  auf  $n+1$ :
  - Füge an einen Baum aus  $n$  Knoten ein Blatt an; über die neue Kante fließen genau 2 Nachrichten
  - Also:  $m(n+1) = m(n) + 2 = 2n - 2 + 2 = 2(n+1) - 2$  ✓

- wo genau liegt der Fehler?
- korrekter Wert der Nachrichtenkomplexität  $\rightarrow$  nächste Folie!

# Nachrichtenkomplexität

(1) Explosionsphase:  $n-2+k$

Anzahl der Initiatoren

- es gibt  $k-1$  Begegnungskanten von Explosionsnachrichten

(2) Kontraktionsphase:  $n$

- über alle Kanten eine Nachricht, nur über die Zentrums-kante zwei

(3) Informationsphase:  $n-2$

- keine Nachricht über die Zentrums-kante

---

$$\Sigma = 3n + k - 4 \quad (\text{mit Information aller Knoten})$$

$$\rightarrow 3(n-1) \text{ für } k=1$$

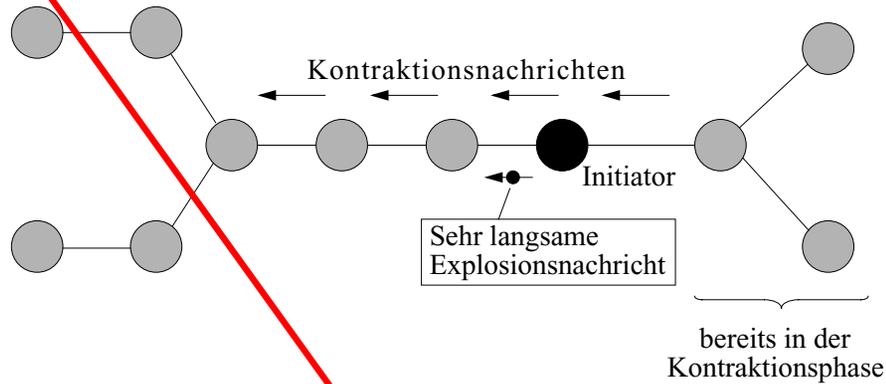
$$\rightarrow 4(n-1) \text{ für } k=n$$

---

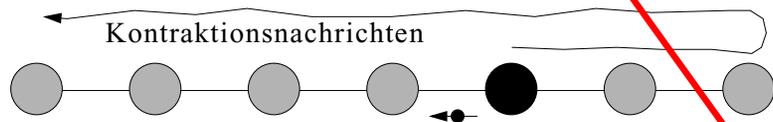
- Wesentlich effizienter als Ringe!
- Wieso? (Ringe sind symmetrischer!)
- Wieso Verfahren nicht "einfach" auf Ringe anwenden?

(eine Kante entfernen)

# Schaden Nachrichtenüberholungen?



- Kann eine Kontraktionsnachricht eine Explosionsnachricht überholen?
- Muss man das vermeiden?
- Lassen sich vielleicht sogar z.T. Nachrichten durch Zusammenfassen (Explosion / Kontraktion) sparen?

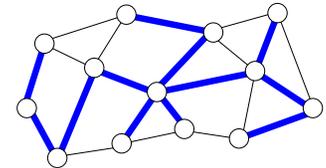


- Wie würde in diesem Fall das Ende erkannt?

# Election auf allgemeinen Graphen

zusammenhängend

- Wieso versagt folgende einfache Idee für allg. Graphen?
  - verwende den Echo-Algorithmus, um einen (einigen) Spannbaum zu konstruieren
  - führe dann Election auf diesem Baum aus



- Wie wäre es damit:
  - jeder Initiator startet seinen eigenen Echo-Algorithmus, mit dem er (über die Echo-Nachrichten) die grösste Identität erfährt
  - jeder Initiator weiss somit, ob er der grösste ist oder nicht und kennt auch den grössten
  - vgl. dies mit dem "bully-Algorithmus" für Ringe: jeder macht einen vollständigen (!) Ringdurchlauf und prüft dabei, ob er der grösste ist
  - ist das korrekt?
  - effizient?

# "Echo-Election" für allgemeine Graphen

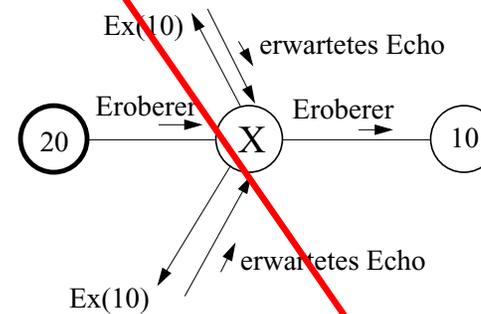
zusammenhängend mit bidirekt. Kanten

- *Generelle Idee*: Chang / Roberts-Algorithmus (also "message extinction"), jedoch *Echo-Algorithmus* anstelle des zugrundeliegenden Ring-Verfahrens
- Also:
  - Jeder Initiator startet "seinen" Echo-Algorithmus
  - Explorer und Echos führen die Identität des Initiators mit
  - Schwächere Nachrichten (Explorer und Echos) werden "verschluckt" (d.h. nicht weitergegeben)
  - Stärkste Welle setzt sich überall durch (und informiert so neben dem Gewinner auch alle Verlierer)
  - Alle anderen Wellen stagnieren irgendwo endgültig (zumindest der Gewinner sendet keine Echos für diese Wellen --> kein anderer Initiator bekommt jemals ein Echo)

- *Veranschaulichung*: "Gleichzeitiges" Einfärben des Graphen mit verschieden dominanten Farben
- Vermutung bzgl. der worst-case und der average-case Nachrichtenkomplexität?

# Varianten, z.B. "Adoption"

- Idee: Stärkerer Knoten ("Eroberer") läuft nicht besiegt Explorern hinterher, sondern "adoptiert" dessen Echos



Beispiel: Knoten X wird erst von 10, dann von 20 erobert. Eroberer-Nachrichten (= Explorer des stärkeren) laufen "direkt" in Richtung des gegnerischen Initiator-knotens

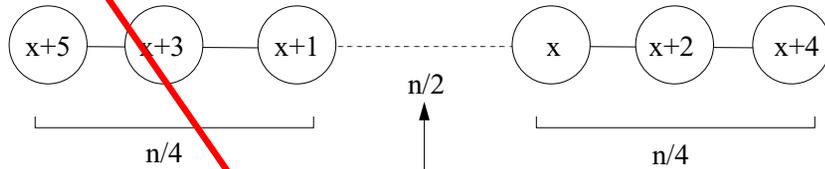
- Kommt statt erwartetem Echo dann ein fremder Eroberer:
  - Fremder > : verloren, dieser erobert nun...
  - Fremder < : Jetzt doch Explorer in diese Richtung senden, da der "Vasall" offenbar nicht stark genug war, um sich durchzusetzen

- *Eigenschaften* dieser Variante:
  - Nur ein einziges Echo pro Knoten (das ist oft praktisch!) (Interpretation: "Verschiedene" Echo-Wellen vereinigen sich)
  - Nicht jeder Knoten wird über seinen Misserfolg informiert (Gewinner kann aber über "seinen" Baum eine Informationswelle starten)

- Es sind noch einige andere Varianten denkbar...

# Nachrichtenkomplexität von Echo-Election

- Worst-case Nachrichtenkomplexität ist  $O(n^2)$



Strecke kann u.U. von jedem der  $n/2$  skizzierten Initiatoren durchlaufen werden (wenn diese geeignet zeitversetzt "zünden")

- Mittlere Nachrichtenkomplexität ist  $O(e \log k)$

Anzahl der Initiatoren

Hier nur Beweisskizze:

- Ein Knoten wird im Mittel  $H_k \approx \log k$  mal erobert ( $\rightarrow$  Anzahl der Rekorde!)
- Eroberter Knoten sendet  $e/n$  Nachrichten im Mittel (Explorer und Echos)  
 $\rightarrow n H_k (e/n) = e H_k$  Nachrichten insgesamt.

## - Bemerkungen:

- Empirische Untersuchungen zeigen, dass die Adoptionsvarianz bei typischen Graphen und mehreren Initiatoren ca. 30% - 50% der Nachrichten spart. (Bei nur einem Initiator sind die Verfahren identisch!)
- Es gibt Verfahren mit geringerer worst-case Nachrichtenkomplexität, diese sind allerdings um einiges aufwendiger!

# Nachrichtenkomplexität: untere Schranke

**Satz:** Die Nachrichtenkomplexität für das Election-Problem in allg. Graphen beträgt mindestens  $\Omega(e + n \log n)$

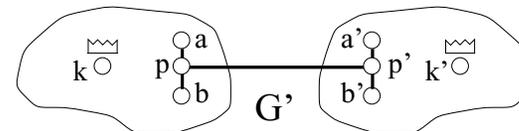
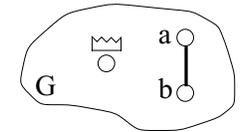
was dominiert typischerweise?

**Beweis:**  $n \log n$  ist untere Grenze, da der Satz auch für Ringe gilt (vgl. dortigen Satz ohne Beweis)

Also noch zu zeigen:  $e$  ist untere Schranke

*Widerspruchsbeweis:* Angenommen, es gäbe einen Election-Algorithmus A, der weniger als  $e$  Nachrichten für einen Graphen G benötigt  $\implies$  es gibt eine Kante  $\overline{ab}$ , über die keine Nachricht fließt

Konstruiere dann  $G'$  aus zwei Kopien von G (mit unterschiedlichen Identitäten aber der gleichen relativen Ordnung) so, dass diese mit einer Kante  $\overline{pp'}$  verbunden werden ( $p$  bzw.  $p'$  werden in die unbenutzten Kanten  $\overline{ab}$  bzw.  $\overline{a'b'}$  neu eingefügt)



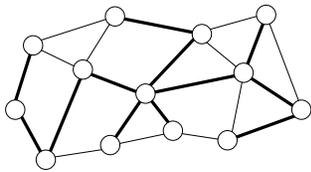
Da zwischen den beiden Teilen keine Nachricht ausgetauscht wird, gewinnen u.U. zwei Prozesse  $k, k'$ !

Beachte bei diesem Beweis:

- bei Anwendung von Algorithmus A auf  $G'$  kann sich jeder Knoten genauso wie der entsprechende im Graphen G verhalten
- die Knoten haben (weder in G noch in  $G'$ ) ein "globales Wissen": sie kennen nicht die Struktur oder Gesamtgröße des Graphen, sie kennen nicht die Identitäten ihrer Nachbarn
- die Knoten wissen insbesondere nicht, ob Situation G oder  $G'$  vorliegt
- Knoten  $p$  und  $p'$  seien keine Initiatoren
- bzgl. der Knotenidentitäten wird nur vorausgesetzt, dass auf diesen eine lineare Ordnung definiert ist (nicht, dass es sich um Nummern handelt)

# Verteilte Spannbaumkonstruktion?

- Gegeben: Zusammenhängendes Netz (mit bidir. Kanten)
- Alle Knotenidentitäten seien verschieden
- Bestimmung eines spannenden Baumes ist oft wichtig:



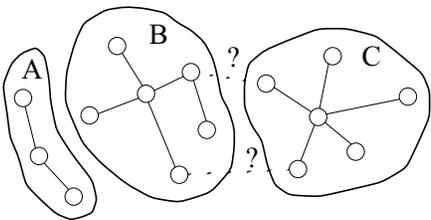
- z.B. um entlang dieses Baumes Information zu verteilen und einzusammeln
- Routing ist wesentlich einfacher, wenn Zyklen ausgeschlossen sind

**Problem:** - Wer initiiert die Spannbaumkonstruktion?

- Basteln alle Initiatoren am gleichen Baum?  
(Problem: Zyklenbildung trotz Dezentralität vermeiden!)

- **Lösung 1:** Election, Gewinner startet Echo-Algorithmus  
(bzw. Variante Echo-Election: Die Echos bilden bereits einen Spannbaum)

- **Lösung 2:** Dezentral werden Fragmente gebildet, die wachsen und sich nach und nach vereinigen



- Initial: Jeder Knoten ist ein Fragment
- Algorithmus von Gallager, Humblet, Spira (GHS) 1983 mit  $2e + 5n \log n$  Nachrichten (im Detail nicht ganz einfach, hier nicht weiter behandelt)
- Problem: Sparsam mit Nachrichten umgehen!

# Election und Spannbaumkonstruktion

*Beh.:* Election und Spannbaumkonstruktion sind in allgemeinen Netzen vergleichbar schwierige Probleme:

*Präziser:* Sei  $C_e$  die Nachrichtenkomplexität des Election-Problems, und  $C_t$  diejenige des Spannbaum-Problems:

- (a) Es gilt für  $C_t$ :  $C_t \leq C_e + 2e$  (wg. obiger "Lösung 1")
- (b) Es gilt für  $C_e$ :  $C_e \leq C_t + O(n)$  (wg. Kplx. Baumelection)

*Interpretation:*

- (a) Hat man mittels Election einen "leader" bestimmt, lässt sich ein eindeutiger Spannbaum einfach ermitteln
- (b) Hat man einen Spannbaum, dann lässt sich ein "leader" einfach (d.h. effizient) bestimmen

Hierbei wird  $2e$  bzw.  $O(n)$  als "klein" gegenüber  $C_e$  und  $C_t$  angesehen

Aus der unteren Schranke  $\Omega(e + n \log n)$  für die Nachrichtenkomplexität des Election-Problems folgt aus (b):

*Das Spannbaum-Problem hat eine Nachrichtenkomplexität von mindestens  $\Omega(e + n \log n)$*

Denn sonst: Konstruiere den Spannbaum effizienter und löse mit zusätzlichen  $O(n)$  Nachrichten das Election-Problem!

--> Der genannte GHS-Algorithmus ist grössenordnungsmässig optimal!

# Anonyme Netze

- Keine Knotenidentitäten
  - bzw.: alle (oder mehrere) *identische* Identitäten
  - bzw.: ggf. vorhandene Knotenidentitäten werden nicht benutzt
- Frage: Was geht dann noch? (Insbes. Symmetriebrechung!?)

- Es gilt: Falls Election in anonymen Netzen geht, dann können die Knoten individuelle Namen bekommen

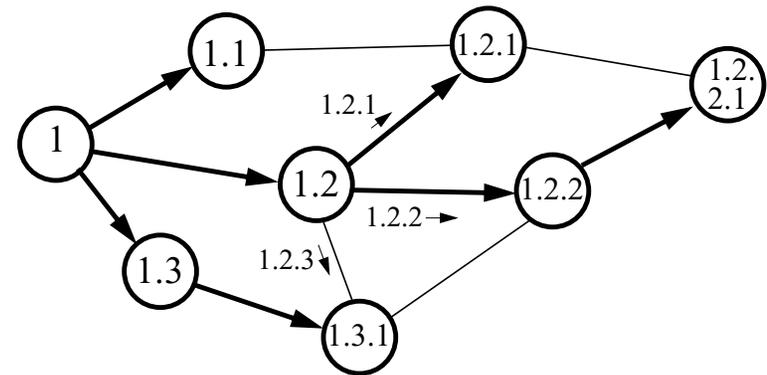
...und *alles* geht wie gehabt!

- 1. Idee: Leader gibt sich einen (neuen) Namen; restliche Knoten führen unter sich eine neue Election durch
- 2. Idee: Leader fragen, wie man heissen soll; dieser denkt sich Namen aus
- Konkretisierung bzw. Variante der 2. Idee:
  - es sei also die Existenz eines "Leaders" angenommen
  - dieser kann mit dem Echo-Algorithmus einen Spannbaum bilden (das klappt auch bei anonymen Knoten!)
  - die Echos melden ("rekursiv") die Grösse (= Anzahl der Knoten) jedes gebildeten Unterbaumes zurück
  - jeder Knoten merkt sich für jede "ausgehende" Kante die Grösse des daran hängenden Unterbaumes
  - nach Konstruktion des Baumes wird (ausgehend vom Initiator als Wurzel) der Baum durchlaufen - dabei wird jedem Unterbaum ein disjunktes Intervall natürlicher Zahlen passender Grösse zugeordnet

- Obiger Satz lässt vermuten, dass Election in anonymen Netzen *nicht* geht, sonst wäre Anonymität kein *grundsätzliches* Problem!

# De-Anonymisierung

Es geht auch direkter ohne explizite Baumkonstruktion, indem der Leader das Netz mit verschiedenen benannten Nachrichten flutet und jeder Knoten die Identität der ersten Nachricht übernimmt, die ihn erreicht - dazu numeriert ein Knoten seine Kanäle bzw. Nachrichten durch und konkateniert seine eigene Identität zu dieser Nummer  
 ==> Alle Nachrichten und damit alle Knoten heissen verschieden!



- Einziger (!) Initiator gibt sich selbst einen Namen "1"
- Jeder Prozess numeriert seine Ausgangskanäle 1,...,k
- Jede von Prozess X über Kanal j versendete Nachricht bekommt zusätzlich den Namen "X.j" dazugepackt
- Ein (noch) anonymer Prozess nimmt den Namen der ersten Nachricht als seinen Namen

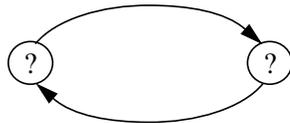
Fragen: - wann ist das Verfahren beendet?  
 - wie erfährt der Initiator dies?

# Election in anonymen Netzen?

- In anonymen Netzen geht manches (z.B. Election?) nicht mehr mit deterministischen Algorithmen --> randomisierte Verfahren helfen gelegentlich!
- Manches geht noch, wenn wenigstens die Knotenzahl bekannt ist (aber auch das hilft nicht immer!)

**Satz:** *Es gibt keinen stets terminierenden Election-Algorithmus für anonymen Ringe*

selbst wenn die Ringgröße den Knoten bekannt ist



- hier für Ringgröße 2
- jeder Knoten befindet sich (bzgl. des Election-Algorithmus) in einem bestimmten Zustand z

- Bew.:**
- Betrachte *Konfigurationen* (hier Quadrupel aus den Zuständen der beiden Knoten und Kanäle)
  - Kanalzustand = Nachrichten, die unterwegs sind
  - Anfangskonfiguration des Algorithmus ist *symmetrisch*:  $(z, z, \emptyset, \emptyset)$ , wegen Anonymität
  - Alle lokalen Algorithmen sind definitionsgemäss identisch --> Knoten können sich jeweils gleich (quasi "zum gleichen Zeitpunkt") verhalten
  - Konfigurationsfolge kann daher aus lauter symmetrischen Konfigurationen bestehen:  $(z, z, \emptyset, \emptyset) \rightarrow (z', z', k, k) \rightarrow \dots \rightarrow \text{etc.}$
  - Falls die Folge endlich ist, könnte der Endzustand symmetrisch sein --> keine Symmetriebrechung möglich!

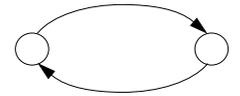
# Ein probabilistischer Election-Algorithmus

```

state := undecided;
while state = undecided do
{ mine := random(0,1);
  send <mine> to neighbor;
  receive <his>;
  if (mine,his) = (1,0) then state := win;
  if (mine,his) = (0,1) then state := lose;
}
    
```

zufälliges Ergebnis 0 oder 1

- hier: *anonyme* Ringe der Größe 2

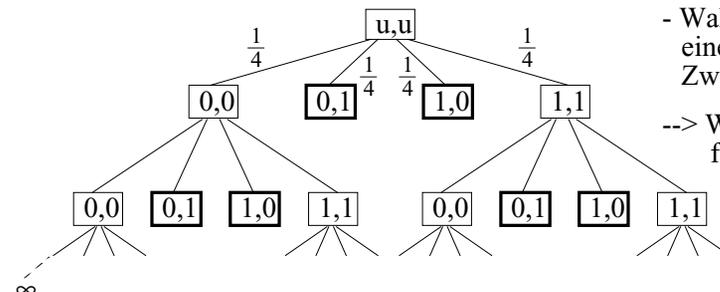


- jeder Knoten führt den gleichen nebenstehenden Algorithmus aus
- Verallgemeinerung auf grössere Ringe?

- Geht es nicht auch mit einem *deterministischen* Algorithmus?

- Beachte: "*korrekt*"  $\neq$  "*korrekt mit Wahrscheinlichkeit 1*"!

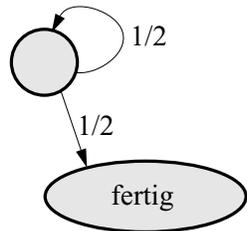
- "*korrekt*" heisst: Algorithmus *hält stets* und liefert richtiges Ergebnis
- obiger Algorithmus ist aber (in diesem Sinne) *nicht korrekt*, da es nicht terminierende Konfigurationsfolgen wie etwa
  - $(0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (0,0) \dots$  oder
  - $(0,0) \rightarrow (1,1) \rightarrow (0,0) \rightarrow (1,1) \rightarrow (0,0) \rightarrow (1,1) \dots$  etc. gibt!
- alle diese Folgen haben aber die Dichte 0 (wenn "random" gut genug ist...)
  - ==> Algorithmus terminiert mit Wahrscheinlichkeit 1
  - ==> Algorithmus ist "*korrekt mit Wahrscheinlichkeit 1*"!



- Wahrscheinlichkeit eines unendlichen Zweiges = 0
- > Wahrscheinlichkeit für leader = 1

# Mittlere Nachrichtenkomplexität?

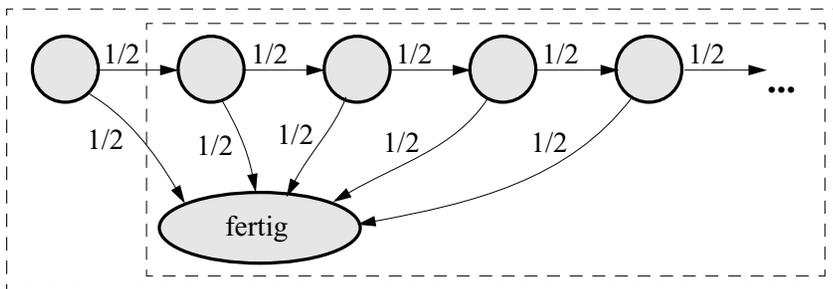
- Beachte: Keine Schranke für *worst-case* Komplexität!
- Die mittlere ("erwartete") Nachrichtenkomplexität lässt sich mittels *Markow-Ketten* leicht ermitteln



- wie hoch ist die erwartete Weglänge  $W$ , wenn mit den angegebenen Wahrscheinlichkeiten (hier jeweils  $1/2$ ) zum Folgezustand verzweigt wird?
- ist dieser Ansatz (gewichtete Summe) korrekt?  

$$1 \frac{1}{2} + 2 \frac{1}{4} + 3 \frac{1}{8} + 4 \frac{1}{16} + \dots = ?$$

- Durch "Aufbröseln" der Schleife ergibt sich diese Darstellung:



- hier ist man mit Wahrscheinlichkeit  $1/2$  nach einem Schritt fertig, oder es liegt (nach einem Schritt) wieder die gleiche Situation vor!
- daraus ergibt sich der *Rekursionsansatz*  $W = 1/2 + 1/2 (1+W)$
- dies liefert  $W=2$  als Lösung
- Bem.: Hinweise zur Varianz etc. findet man in entsprechenden Mathematik-Lehrbüchern

# Probabilistische Algorithmen

- Der klassische "totale" Korrektheitsbegriff von Algorithmen kann auf zweierlei Weise abgeschwächt werden:

## 1. Sogenannte *Las Vegas-Algorithmen*:

- Abschwächung der Terminierungsforderung
- also: "Partiell korrekt und Terminierung mit *Wahrscheinlichkeit 1*"
- beachte: die (worst-case) Laufzeit solcher Algorithmen ist unbeschränkt!
- Beispiel: obiger Election-Algorithmus für anonyme Ringe

## 2. Sogenannte *Monte Carlo-Algorithmen*:

- Abschwächung der partiellen Korrektheit
- "terminiert stets, ist aber nur mit Wahrscheinlichkeit  $p < 1$  partiell korrekt"
- also:  $\exists$  *Restwahrscheinlichkeit*  $\epsilon = 1-p > 0$ , dass das Ergebnis falsch ist!
- nur verwenden, wenn:
  - $\epsilon$  sehr *klein* ist (oft: als Parameter des Algorithmus, etwa abhängig von der Laufzeit und damit "beliebig klein" wählbar)
  - dadurch deutliche Vorteile erzielbar (Problem effizienter oder überhaupt erst lösbar)
- beachte den "Sonderfall"  $p=1$  (also  $\epsilon=0$ ): ein solcher Monte Carlo-Algorithmus wäre *total korrekt* (hält stets und das Ergebnis ist dabei ("mit Wahrscheinlichkeit 1") korrekt)!

# Las Vegas-Election-Algorithmus für anonyme Ringe bekannter Grösse

- 1981 von Itai/Rodeh: Basiert auf Chang/Roberts-Verfahren

- Prinzip:

oder ein anderer Wert, z.B.  
 $2n$ ,  $n^2$  oder einfach 2 ?

- wähle eigene Identität  $id = \text{random}(1, \dots, n)$ , mit  $n = \text{Ringgrösse}$
- message extinction wie gehabt
- Nachrichten enthalten einen "hop counter": Zählt Anzahl besuchter Knoten
- falls eine Nachricht mit eigener Identität empfangen wird:
  - prüfe, ob hop counter =  $n$ 
    - *nein* -->  $\exists$  anderen Knoten gleicher Identität (merken mittels Flag!)
    - *ja* --> gewonnen! (aber falls Flag gesetzt, gibt es andere Gewinner!)
- falls es mehrere Gewinner gibt:
  - nur diese führen eine *neue Election-Runde* durch
  - daher enthalten Nachrichten auch eine Rundenkennung (alte Nachrichten werden in der nächsten Runde einfach ignoriert)

FIFO-Kanäle  
notwendig?

---

- Ohne Beweis: *Erwartungswert bzgl. Rundenzahl*  $\leq e(n/n-1)$

- vgl. mit vorherigem Algorithmus für Ringgrösse 2

2.718281828...

- Berechnungsdauer ist im Prinzip aber unbegrenzt!

- Algorithmus ist partiell korrekt: *Wenn* er hält, dann mit genau einem leader!

- Verallgemeinerung auf *allgemeine Netze* mit Echo-Algorithmus (statt Ring) ist möglich:

- wenn die durch Echos gemeldete Baumgrösse  $\neq n$  ist, neue Runde starten etc.