

Wahlalgorithmen

- ▶ Überblick
- ▶ Wahl auf Ringen
 - ▶ Chang-Roberts Algorithmus
 - ▶ Peterson Algorithmus
- ▶ Wahl auf Bäumen
- ▶ Wahl auf beliebigen Topologien

Wahlalgorithmen: Überblick

Formale Anforderungen an einen Wahlalgorithmus

- WA1** Eindeutigkeit: Es darf keine zwei Knoten P_1, P_2 im System geben, die sich beide als Anführer betrachten
- WA2** Terminierung: In endlicher Zeit gibt es einen Knoten P_i , der sich als Anführer betrachtet

Wahlalgorithmen: Überblick

Problemstellung

- ▶ In verteilten Systemen kann es viele Situationen geben, die einen ausgewählten Knoten erfordern („Anführer“)
 - ▶ Koordinator von verteilten Aktionen
 - ▶ Erzeugung von eindeutigen Token im System
 - ▶ ...
- ▶ Man möchte diesen Knoten automatisch bestimmen lassen
 - ▶ Initial bei Start des Systems
 - ▶ Zur Neukonfiguration des Systems nach Fehlern

Wahlalgorithmen: Überblick

Varianten

- ▶ Alle Knoten erfahren vom Ende oder vom Ergebnis der Wahl
- ▶ Verschiedene Topologien: Ringe, Bäume, beliebige Netze
- ▶ Variable/unbekannte oder feste/bekannte Knotenanzahl?
- ▶ Nicht unterscheidbare Knoten oder Knoten mit eindeutigen IDs?

Erforderliche Randbedingung (für deterministische Wahl)

- ▶ Es muss möglich sein, die einzelnen Knoten voneinander zu unterscheiden (z.B. eindeutige Netzadresse)

Literatur zu Wahlalgorithmen

Wahl auf Ringen

- ▶ N. Lynch: Distributed Algorithms. Morgan Kaufmann Pub., 1996
- ▶ G. L. Lann: Distributed Systems - Towards a Formal Approach. In Information Processing 77, pp 155-160, IFIP, North-Holland, 1977
- ▶ E. Chang, R. Roberts: An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes, Comm. ACM, 22(5): 281-283, Mai 1979
- ▶ G. L. Peterson: An $O(n \log n)$ unidirectional distributed algorithm for the circular extrema problem. ACM Transactions on Programming Languages and Systems, pp 758-762, Oct. 1982

Wahl auf Bäumen und auf beliebigen Graphen

- ▶ F. Mattern: Verteilte Basisalgorithmen. Springer-Verlag, 1989

Chang-Roberts-Algorithmus

Voraussetzung:

- ▶ Jeder Knoten besitzt eine eindeutige ID , auf welcher eine totale Ordnung definiert ist.

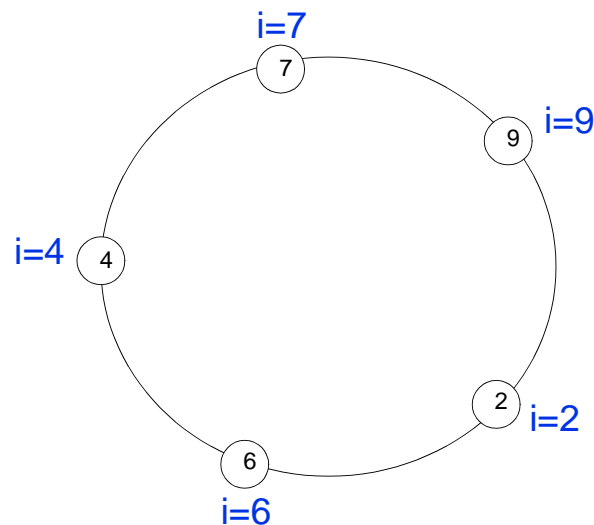
Ziel:

- ▶ Finde Knoten mit maximaler ID

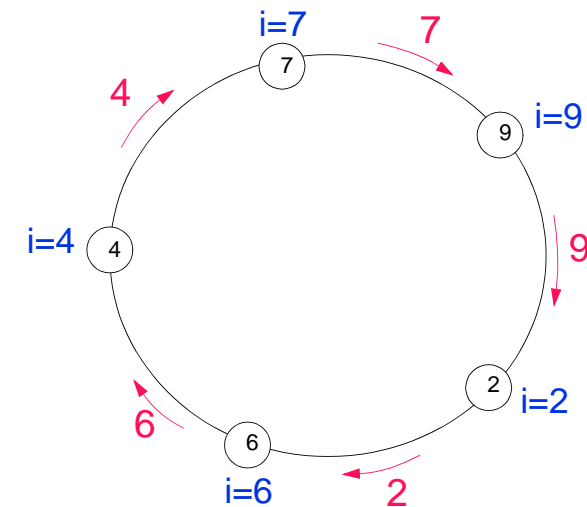
Ablauf

- ▶ Beim Start des Wahlalgorithmus (in allen Knoten): Sende eigene Knoten-ID im Ring weiter, Knoten wird wählbar
- ▶ Bei Empfang einer $ID i_{rx}$:
 - ▶ Falls $i_{rx} >$ eigene ID : i_{rx} weitersenden; Knoten nicht mehr wählbar
 - ▶ Falls $i_{rx} =$ eigene ID : Falls wählbar: als Anführer gewählt (sonst: Nachricht ignorieren)
 - ▶ Falls $i_{rx} <$ eigene ID : Nachricht ignorieren

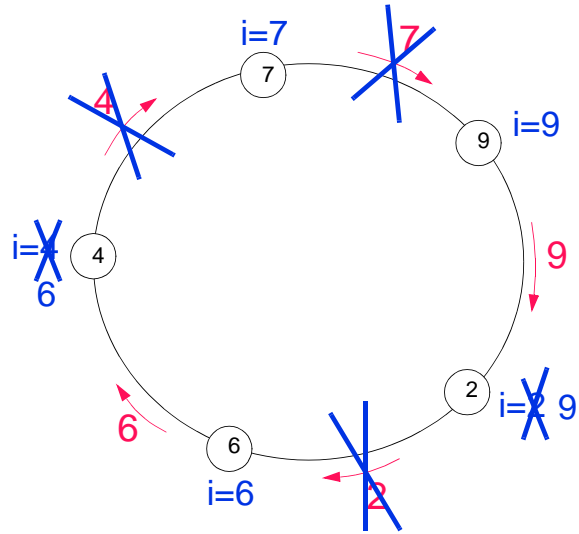
Chang-Roberts-Algorithmus



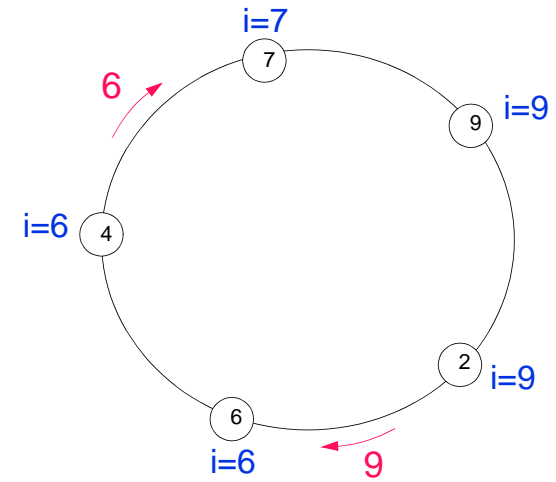
Chang-Roberts-Algorithmus



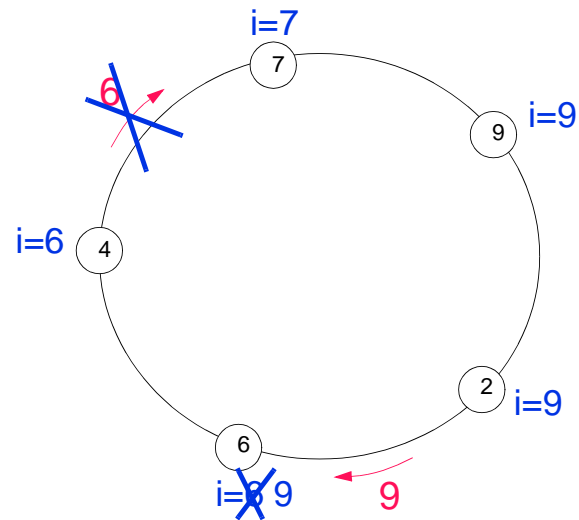
Chang-Roberts-Algorithmus



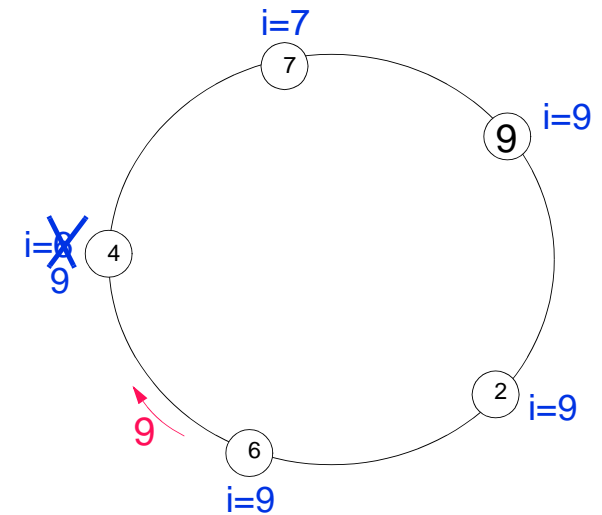
Chang-Roberts-Algorithmus



Chang-Roberts-Algorithmus



Chang-Roberts-Algorithmus



Chang-Roberts-Algorithmus

Eigenschaften

- ▶ Nach spätestens N Schritten gibt es einen Knoten, der sich für den Anführer hält
- ▶ Kein anderer Knoten hält sich je für den Anführer

Erweiterung

- ▶ Sobald ein Knoten als Anführer bestimmt ist, kann er dies durch spezielle Mitteilungs-Nachrichten allen anderen Knoten im Ring mitteilen
- ▶ Erst mit dieser Erweiterung wird eine Terminierung des Algorithmus erreicht (alle anderen Knoten warten sonst weiterhin auf Nachrichten)

Chang-Roberts-Algorithmus

Begründung der Korrektheit – Eindeutigkeit

- ▶ Initial wird eine Nachricht mit einer bestimmten ID nur vom Knoten mit dieser ID erzeugt
- ▶ Diese ID wird von weiteren Knoten nur dann weitergereicht, wenn deren eigene ID kleiner ist. Insbesondere: Der Knoten mit der größten ID reicht keine anderen IDs weiter
- ▶ Ein Knoten kann nur dann gewählt werden, wenn dessen ID alle Knoten durchlaufen hat
- ▶ Dies ist nach dem zweiten Punkt nur für die größte vorhandene ID möglich, es kann also maximal einen Anführer geben

Chang-Roberts-Algorithmus

Begründung der Korrektheit – Terminierung

- ▶ Eine Nachricht mit der größten ID wird gemäß Spezifikation von jedem Knoten weitergereicht.
- ▶ Nach N Nachrichten ist die größte ID wieder beim Ursprungsknoten angekommen; unter der Annahme, dass jede Nachricht in endlicher Zeit beim Empfänger ankommt, ist der Knoten mit größter ID in endlicher Zeit ermittelt

Chang-Roberts-Algorithmus

Komplexitätsbetrachtung

- ▶ Annahmen zur Analyse der Zeitkomplexität
 - ▶ Lokale Aktionen benötigen keine Zeit, gesendete Nachrichten kommen jeweils innerhalb einer maximalen Nachrichtenlaufzeit an
- ▶ Worst-Case-Betrachtung:
 - ▶ Zeit: $O(N)$ Nachrichtenlaufzeiten
 - ▶ Nachrichten: $O(N^2)$
(Man ordnet die IDs absteigend an, dann ergibt sich $\sum_{i=1}^N i = n(n+1)/2$)
- ▶ Betrachtung des mittleren Werts:
 - ▶ Annahme: Alle möglichen Verteilungen der IDs sind gleichwahrscheinlich, IDs mit Werten von $1..N$
 - ▶ Zeit: unverändert $O(N)$
 - ▶ Es kann gezeigt werden: $O(N \log N)$ (Siehe Literatur)

Chang-Roberts-Algorithmus

Weitere Optimierungen möglich?

- ▶ Theoretische Erkenntnis (siehe Literatur): $O(N \log N)$ ist untere Schranke für die Nachrichtenkomplexität!
- ▶ Geht es besser als $O(N^2)$ im worst-case?

Peterson-Algorithmus

Ziel

- ▶ Verringerung der Nachrichtenanzahl im schlechtesten Fall auf $O(N \log N)$

Zustandsvariablen:

- ▶ Zustand: *aktiv*, *weiterleiten*, *gewählt*, initial: *aktiv*
- ▶ ID: eine Knoten-ID, initial: eigene ID

Chang-Roberts-Algorithmus – bidirektional

Einfache randomisierte Variante

- ▶ Jeder Knoten würfelt initial einen Wert aus $0, 1$ und sendet seinen Wert entsprechend zum linken oder zum rechten Nachbarn weiter
- ▶ Normaler Chang-Roberts-Algorithmus wird ausgeführt; alle Nachrichten werden dabei in dem Umlaufsinn weitergereicht, in dem sie bei einem Knoten ankommen

Eigenschaften

- ▶ Worst-case beträgt immer noch $O(N^2)$ Nachrichten, tritt aber mit geringerer Wahrscheinlichkeit auf
- ▶ Mittlere Nachrichtenzahl ist günstiger, aber noch aufwendiger zu berechnen und asymptotisch immer noch $O(N \log N)$

Peterson-Algorithmus

Ablauf in Phasen

In jeder Phase wird die Zahl der aktiven Prozesse mindestens **halbiert**

- ▶ Jeder aktive Prozess i sendet den Wert seiner Variablen ID an die *beiden nächsten aktiven* Prozesse im Ring weiter
- ▶ Jeder aktive Prozess vergleicht seine ID mit der von den beiden aktiven Vorgängern empfangenen ID ;
 - ▶ hat der *direkte* (aktive) Vorgänger die größte ID , so bleibt der Prozess i aktiv, und übernimmt die ID vom Vorgänger
 - ▶ ansonsten wechselt der Prozess in den Zustand *weiterleiten*
- ▶ Wenn die vom direkten aktiven Vorgänger empfangene ID gleich der eigenen ID -Variablen ist, so ist der Knoten zum Anführer gewählt

Peterson-Algorithmus

(Achtung: Variante aus dem Papier)

```

tid = initial value

do forever
begin /* start phase */
  send(tid)

  receive(ntid)
  if (ntid == initial value) then announce elected
  if (tid > ntid) then send(tid)
  else send(ntid)

  receive(nntid)
  if (nntid == initial) value then announce elected
  if (ntid >= max(tid, nntid)) then tid = ntid
  else goto relay
end

relay: /* relay modus */
do forever
begin
  receive(tid)
  if (tid = initial value) then announce elected
  send (tid)
end

```

Peterson-Algorithmus

Nachrichten-Komplexität im Worst-Case:

- ▶ In jede Phase wird die Anzahl der aktiven Knoten mindestens halbiert, daher maximale Anzahl von Phasen: $\lfloor \log N \rfloor + 1$ (Ein aktiver Prozess überlebt die Phase nur wenn sein direkter, aktiver Vorgänger nicht überlebt.)
- ▶ Je Phase werden maximal $2N$ Nachrichten von den aktiven Prozessen versendet
- ▶ Also Nachrichtenkomplexität: $O(N \log N)$

Zeit-Komplexität (naives Vorgehen):

- ▶ $O(\log N)$ Phasen
- ▶ Jede Phase dauert maximal $O(N)$ Nachrichtenlaufzeiten
- ▶ Also Komplexität $O(N \log N)$

Genauere Betrachtung möglich (hier nicht näher untersucht)

- ▶ Komplexität $O(N)$

Anmerkungen zur Wahl auf Ringen

- ▶ Mögliche Probleme in der Praxis
 - ▶ Nachrichtenverluste
 - ▶ Rechnerausfälle
 - ▶ Mehrfache Ausführung
 - ▶ Doppelt vergebene IDs
- ▶ Konsequenz

Einfache algorithmische Idee in einem einfachen Systemmodell muss in der Realität ggf. in komplexere Algorithmen eingebettet werden, um den realen Randbedingungen gerecht zu werden

Wahl auf Bäumen

Grundidee:

Traversieren des kompletten Baumes, um den Knoten mit der größten ID zu ermitteln

- ▶ Sequentielles Traversieren möglich
- ▶ Paralleles Traversieren bietet sich für verteilte Systeme an

Sequentielles Traversieren

Bekanntes *depth first*-Vorgehen wie bei lokalen Datenstrukturen:

```
class Node implements NodeRemote {
    ArrayList<Node> children = ...;
    int id = ...;

    int findMax() {
        int maxVal = id;
        Iterator<Node> it = children.iterator();

        while(it.hasNext()) {
            int val = it.next().findMax();
            if ( val > maxVal )
                maxVal = val;
        }
        return maxVal;
    }
}
```

Paralleles Traversieren

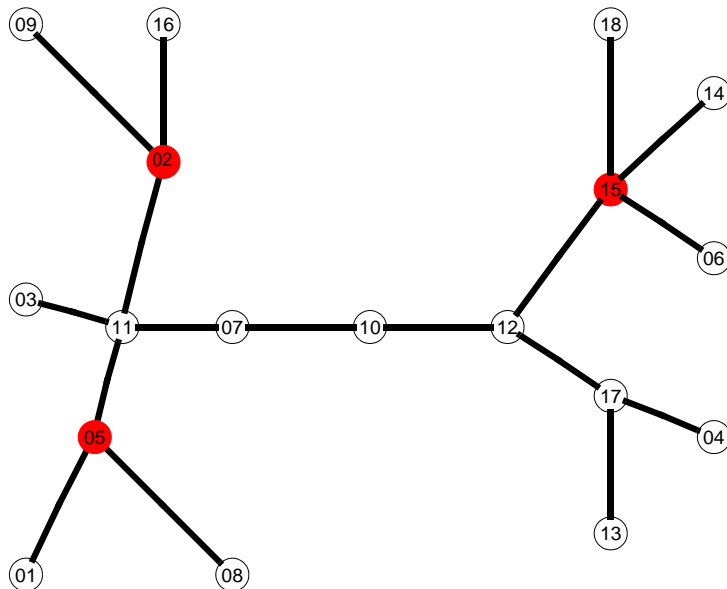
Zeitgewinn durch Parallelisierung

- ▶ Sequentiell: Bei N Knoten $2(N - 1)$ Kommunikationsschritte!

Wellenverfahren

- ▶ *Explosionswelle* durchläuft den Baum bis zu den Blättern
- ▶ *Echowelle* läuft wieder zurück und bestimmt Anführer
⇒ teilweise Überlappung mit Explosionswelle möglich
- ▶ *Informationswelle* kann alle Knoten über das Ergebnis der Wahl informieren

Wahl auf Bäumen

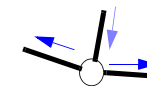


Wahlalgorithmus für Bäume

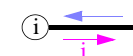
- ▶ Ein Starter sendet Explosionsnachrichten in alle Richtungen



- ▶ Ein innerer Knoten, der erstmals eine Explosionsnachricht empfängt, sendet wiederum Explosionsnachrichten in alle anderen Richtungen aus

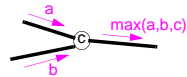


- ▶ Ein Blattknoten, der eine Explosionsnachricht erhält, sendet eine Echonachricht mit seiner ID zurück

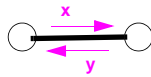


Wahlalgorithmus für Bäume

- ▶ Ein innerer Knoten mit k Kanten, der auf $k - 1$ Kanten Echonachrichten erhalten hat, sendet auf der übrigen Kante eine Echonachricht mit dem Maximum aller ID s

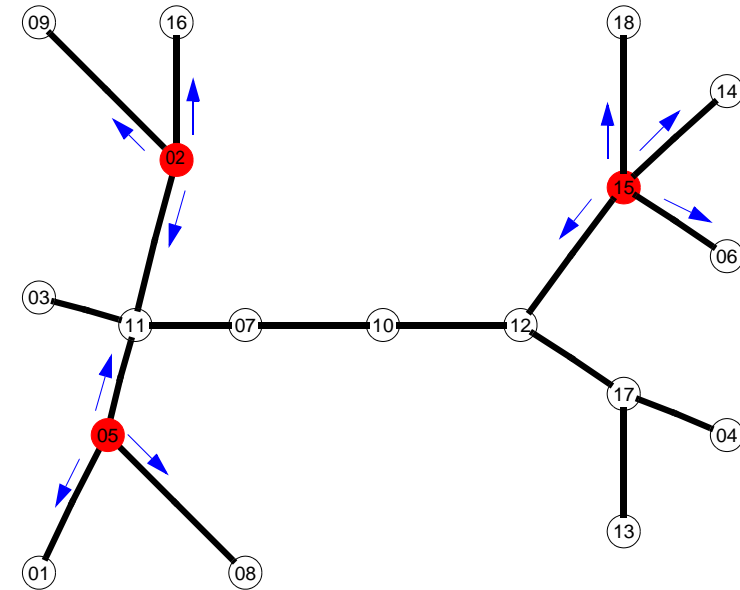


- ▶ Erhält ein Knoten nach Versenden seiner eigenen Echonachricht eine weitere Echonachricht, so ist das Maximum beider ID s die höchste ID -Nummer im Netz

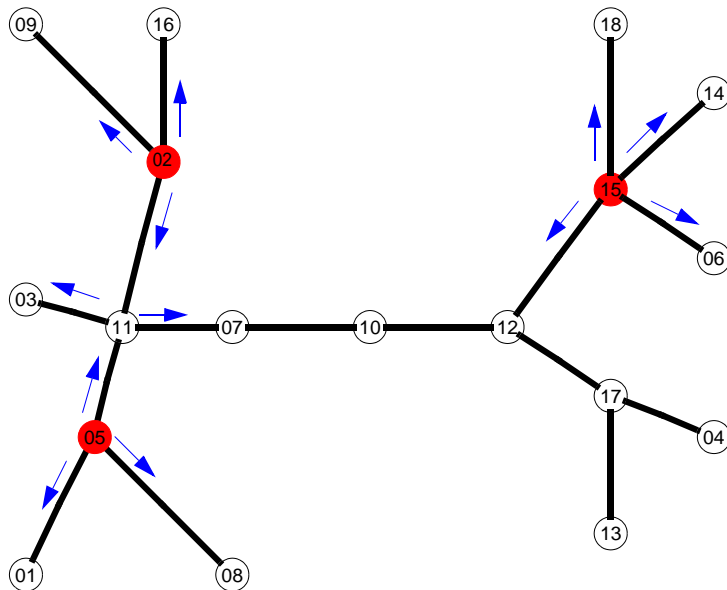


- ▶ Muss lediglich ein beliebiger Anführer bestimmt werden, so kann man jetzt einfach einen der beiden Knoten auswählen
- ▶ Ansonsten kann man in einer Informationswelle alle Knoten über die ermittelte höchste ID informieren

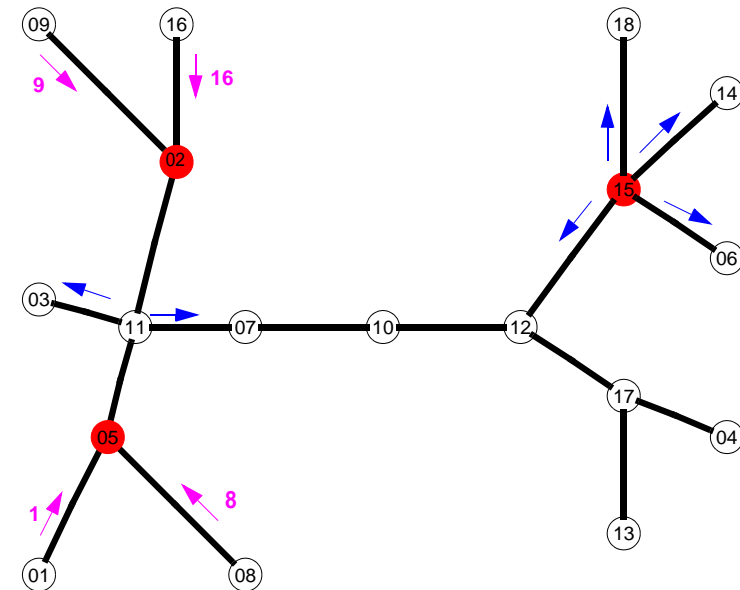
Wahl auf Bäumen



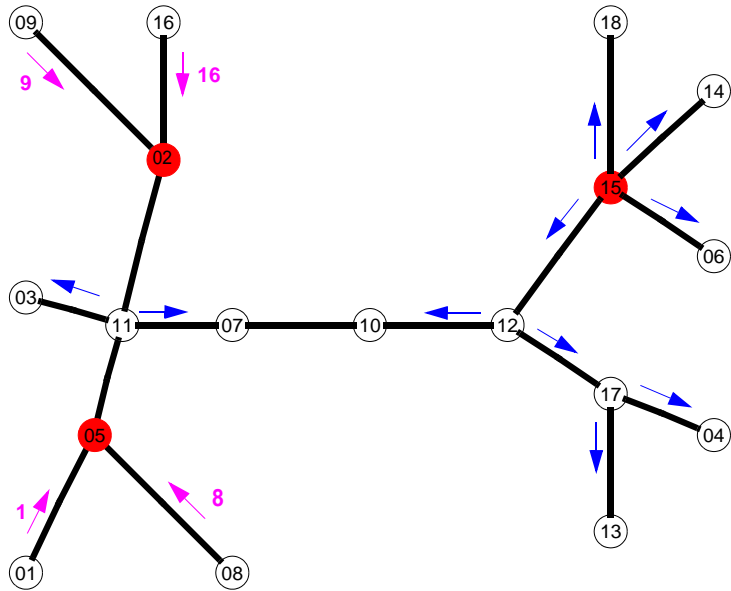
Wahl auf Bäumen



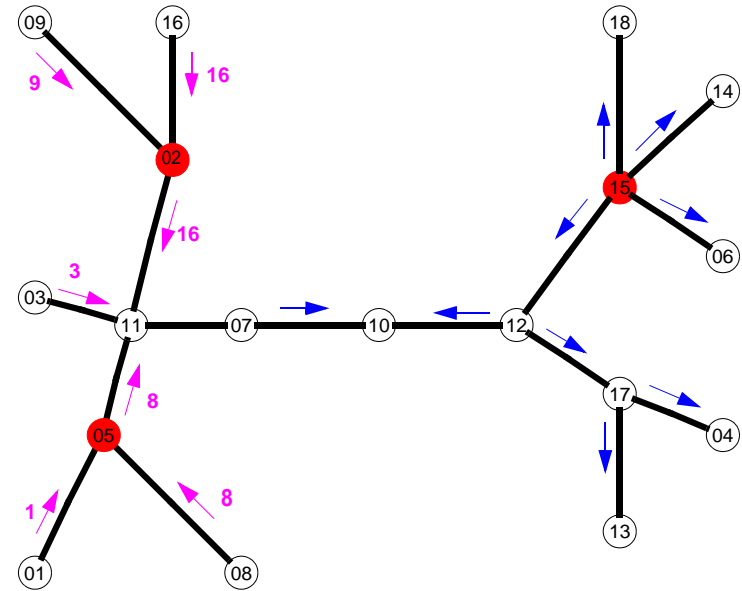
Wahl auf Bäumen



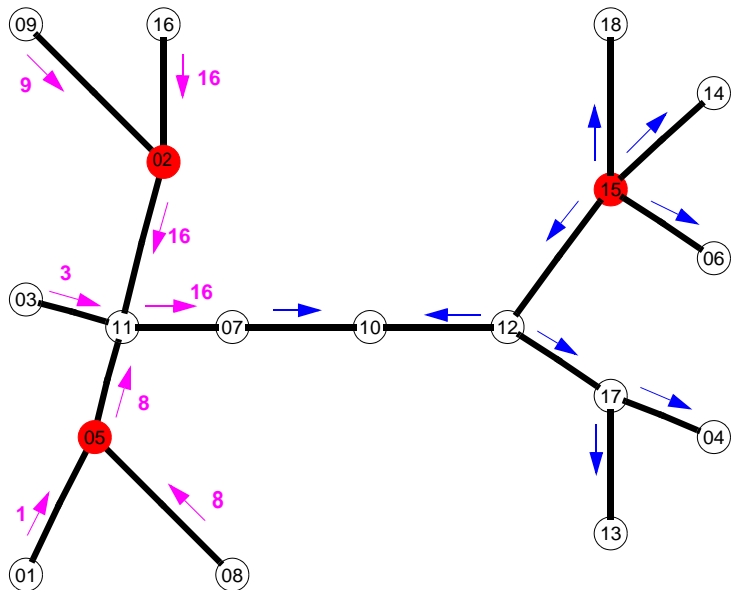
Wahl auf Bäumen



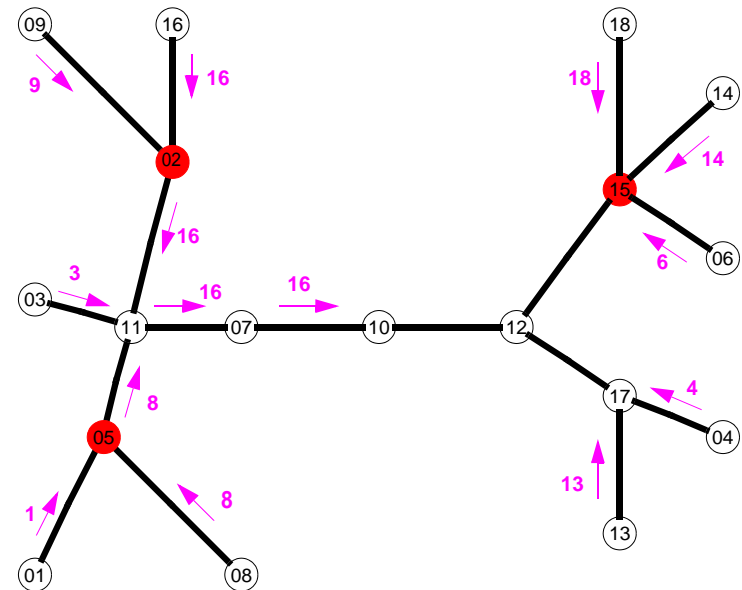
Wahl auf Bäumen



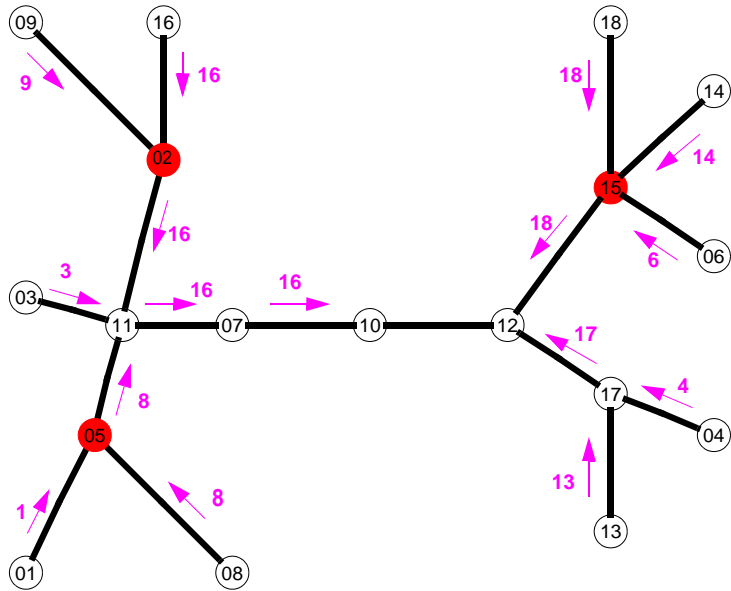
Wahl auf Bäumen



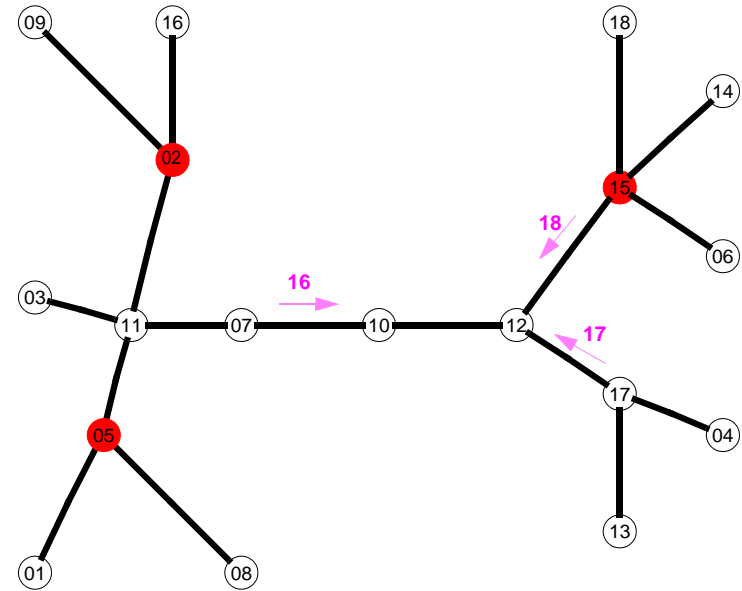
Wahl auf Bäumen



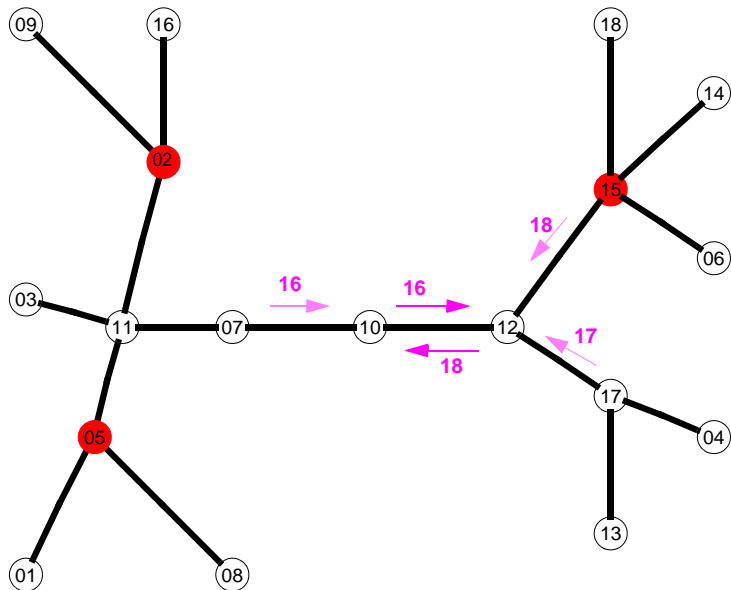
Wahl auf Bäumen



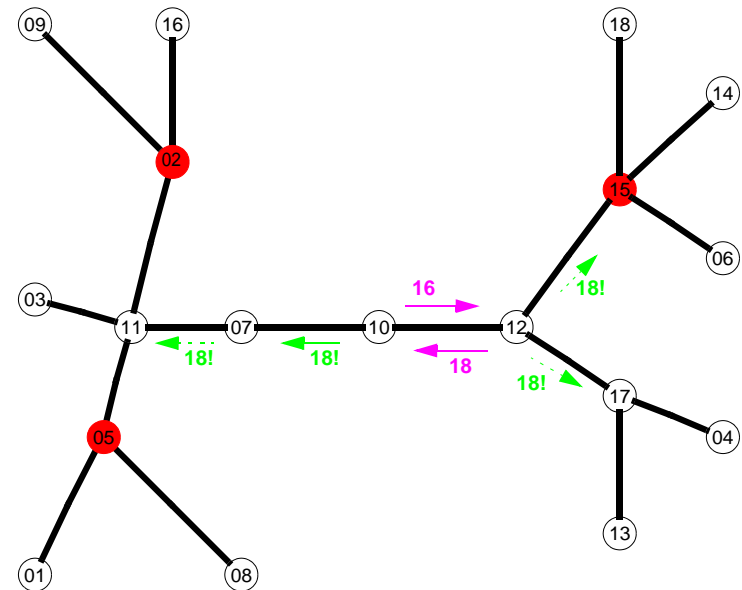
Wahl auf Bäumen



Wahl auf Bäumen



Wahl auf Bäumen



Wahl auf Bäumen: Zusammenfassung

- ▶ Initiierung des Algorithmus durch Start-Welle (Election-Welle)
- ▶ Bildung von gerichteten Baumkanten in einer Echo-Welle; dabei Informationssammlung (z.B. größte ID) möglich
- ▶ Es kann genau eine Kante geben, bei der Echo-Nachrichten in beiden Richtungen laufen
 - ▶ Gewinner kann durch gesammelte Information ermittelt werden
 - ▶ oder: Auswahl eines der beiden Knoten an dieser Kante als Gewinner

Ablauf der sequentiellen Traversierung

Durch Starter generiertes Token durchläuft Graphen nach dem *Depth First*-Prinzip; es kann mehrere Starter geben, aber nur ein einziges Token kehrt zu seinem Starter zurück

- ▶ Beim Starter:
 - ▶ Initial wird ein Token erzeugt (über eine „virtuelle Baumkante“ empfangen)
 - ▶ Wenn das Token vom letzten Nachbar zurückkommt („über die virtuelle Baumkante zurückgeschickt wird“), ist die Wahl beendet

Wahl auf beliebigen Topologien

Sequentielle Traversierung des Netzes

- ▶ Pfadverfahren
- ▶ Kantenfärbungsverfahren

Parallele Traversierung des Netzes

- ▶ Echo/Election-Algorithmus
- ▶ ... und Varianten davon

Konstruktion eines virtuellen Baums auf dem Netz

Ablauf der sequentiellen Traversierung

- ▶ Jeder Baumknoten, beim ersten Empfang des Tokens:
 - ▶ Falls zuvor bereits ein Token mit größerer ID empfangen wurde, wird das Token „verschluckt“
 - ▶ Ansonsten sendet der Knoten das Token zu einem Nachbarn, der das Token bisher noch nicht besessen hat
 - ▶ Wenn es keinen solchen Nachbarn mehr gibt, wird das Token über die Kante zurückgesendet, über die es zuerst empfangen wurde
- ▶ „Erster Empfang des Tokens“:
 - ▶ Mehrfacher Empfang des Tokens über verschiedene Pfade (Schleifen) muss verhindert oder erkannt werden

Weitere Verfahren

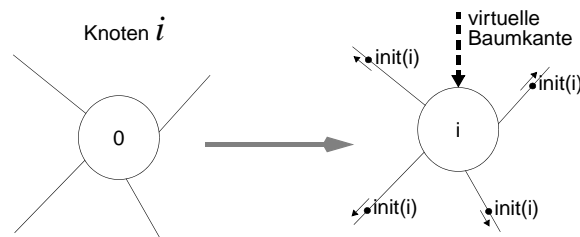
- ▶ Pfadverfahren
 - ▶ Jeder Knoten kennt die Identität seiner Nachbarn, und im Token wird der komplette bisher durchlaufene Pfad gespeichert
 - ▶ Jeder Knoten kann selbst entscheiden, welche Nachbarn das Token noch nicht besessen haben
- ▶ Kantenfärbungsverfahren
 - ▶ Die Knoten müssen die Identitäten der Nachbarn nicht kennen, im Token wird keine Pfadinformation gespeichert
 - ▶ Das Token muss über alle Kanten gesendet werden, um festzustellen, ob der Nachbar das Token bereits besessen hat. Bereits verwendete Kanten werden markiert („gefärbt“), um jede Kante genau einmal zu verwenden

Parallele Traversierung für beliebige Topologien

- ▶ Ziel: Zeitgewinn durch Parallelisierung
- ▶ Grundprinzip basiert auf Echo-Wahlverfahren auf Bäumen
 - ▶ Zusätzlich Erkennung von Schleifen, Elimination von Kanten und dadurch Bildung eines virtuellen Baums
- ▶ Gewinner soll der Initiator-Knoten sein mit der größten ID
- ▶ Hier betrachtete Algorithmen:
 - ▶ Echo/Election
 - ▶ Adoptionsverfahren

Echo/Election-Algorithmus

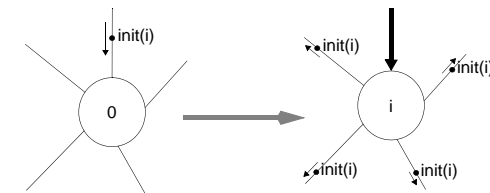
- 1 Ein freier Knoten i wird spontan zum Initiator:



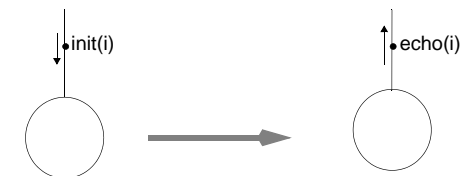
Echo/Election-Algorithmus (Forts.)

- 2 Ein freier Knoten wird von der Nachricht $init(i)$ erreicht:

- a) $Grad > 1$

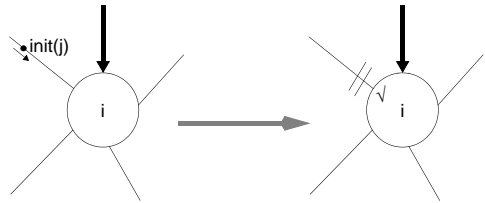


- b) $Grad = 1$



Echo/Election-Algorithmus (Forts.)

- 3) $init(j)$ trifft auf einen mit i markierten Knoten ($Grad > 1$):
- a) $j = i$, d.h. zwei Explorers identischer Markierung begegnen sich

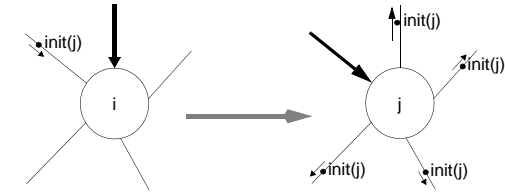


Falls die letzte Nicht-Baumkante als erledigt festgestellt wird, wird ein Echo über die Baumkante verschickt

- b) $j < i$ (d. h. der eintreffende Explorer ist kleiner als der zuletzt versandte): Es erfolgt keine Zustandsänderung! (Nachricht wird *verschluckt*)

Echo/Election-Algorithmus (Forts.)

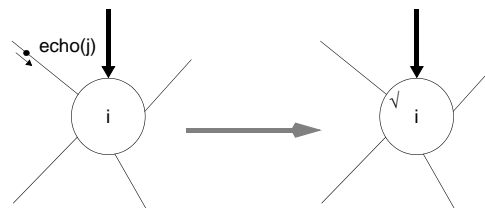
- c) $j > i$ (d. h. Knoten wird von stärkerem Initiator erobert)



Falls ein Initiator erobert wird, entfällt damit dessen virtuelle Baumkante.

Echo/Election-Algorithmus

- 4) Es trifft ein Echo ein:
- a) $i = j$

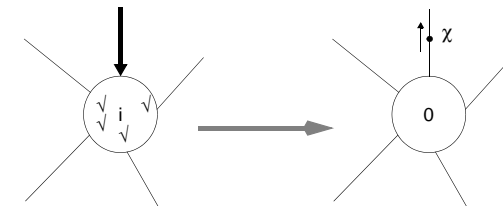


Falls die letzte Nicht-Baumkante als erledigt festgestellt wird, wird ein Echo über die Baumkante verschickt.

- b) $j < i$: Nachricht wird ignoriert

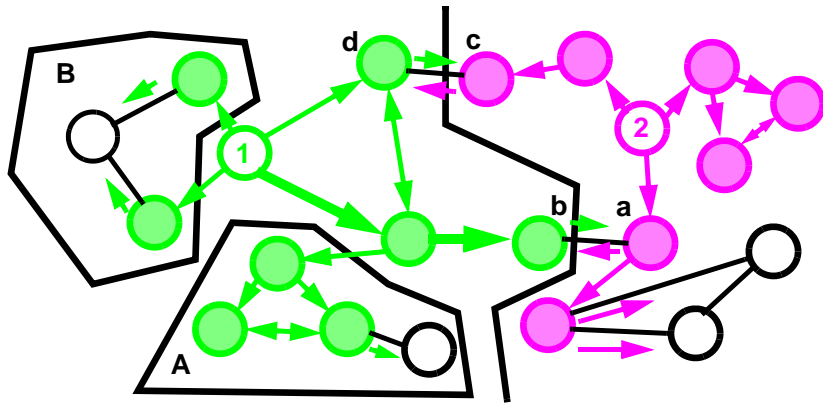
Echo/Election-Algorithmus

- 5) Zurücksenden eines Echos:



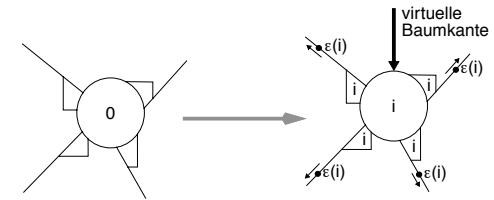
Falls es sich um einen Initiator handelt (d. h. die Baumkante ist virtuell), so ist er der Gewinner.

Echo/Election-Adoptionsverfahren: Idee



Echo/Election-Adoptionsverfahren

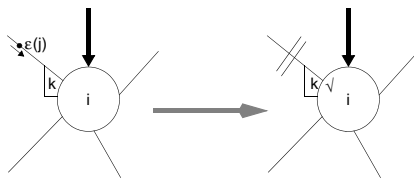
- 1) Ein freier Knoten i wird spontan zum Initiator:
wie bisher + ausgehende Kanten werden mit i markiert



- 2) Ein freier Knoten wird von einem Explorer $\varepsilon(i)$ erreicht:
 $Grad > 1$: wie bisher + ausgehende Kanten werden mit i markiert
 $Grad = 1$: wie bisher

Echo/Election-Adoptionsverfahren (Forts.)

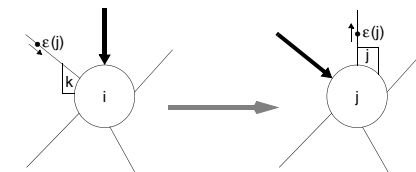
- 3) Ein Explorer $\varepsilon(j)$ trifft auf Knoten mit Markierung i ($Grad > 1$); die Eingangskante sei mit k markiert:
 - a) $k = j$, d. h. zwei Explorer gleicher Markierung begegnen sich



- b) $k > j$, d. h. der eintreffende Explorer ist kleiner als der in Gegenrichtung zuletzt versandte: keine Zustandsänderung

Echo/Election-Adoptionsverfahren (Forts.)

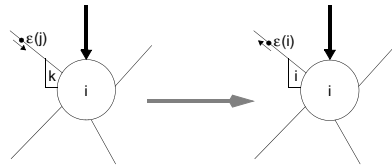
- c) $j > i$ (damit natürlich auch $j > k$): **Adoption**: Explorer wird über die bisherige Baumkante gesendet



Falls ein Initiator erobert wird (dessen virtuelle Baumkante damit entfällt), wird ein ggf. Echo zurückgesandt

Echo/Election-Adoptionsverfahren (Forts.)

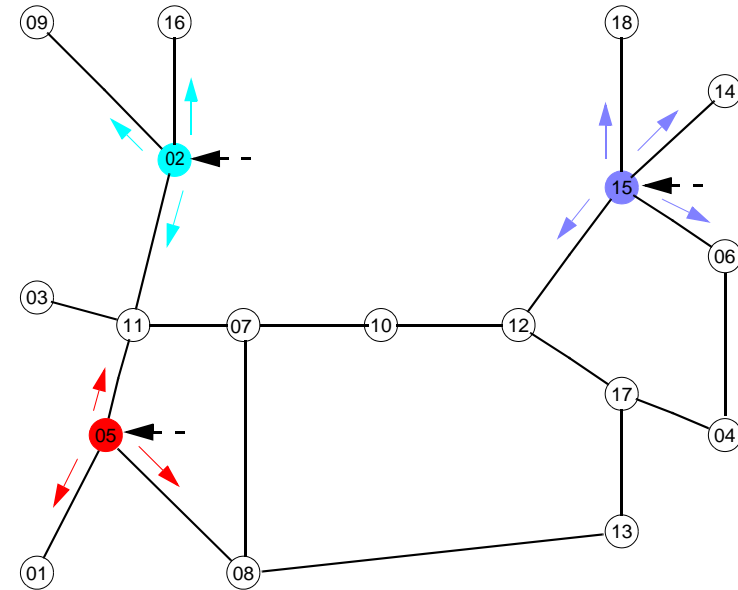
d) $i \geq j > k$



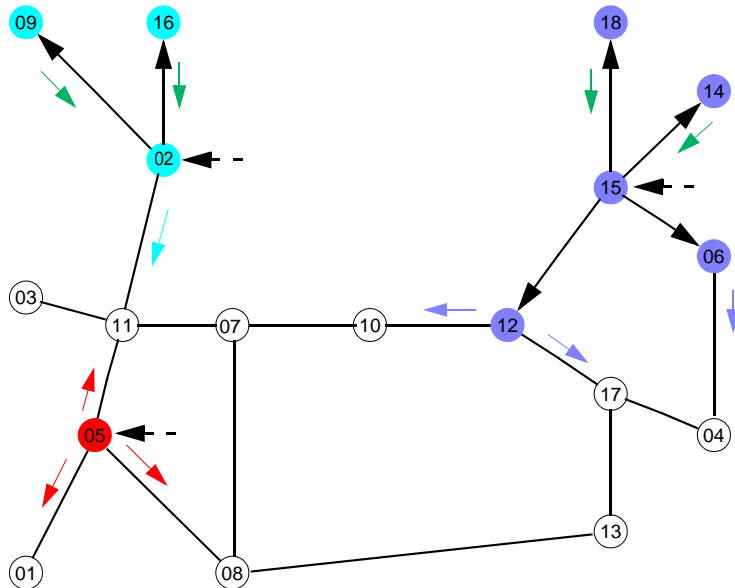
- Falls $i = j$ ist, gilt die Baumkante als erledigt.
 Dieser Fall tritt auf wenn es mind. 3 Initiatoren gibt.
- 1 Ursprünglich gehörte das Gebiet k
 - 2 Es wurde von i erobert
 - 3 Nun will es j übernehmen es gilt aber $i \geq j > k$

- 4) Es trifft ein Echo ein:
wie bisher
- 5) Zurücksenden eines Echo:
wie bisher

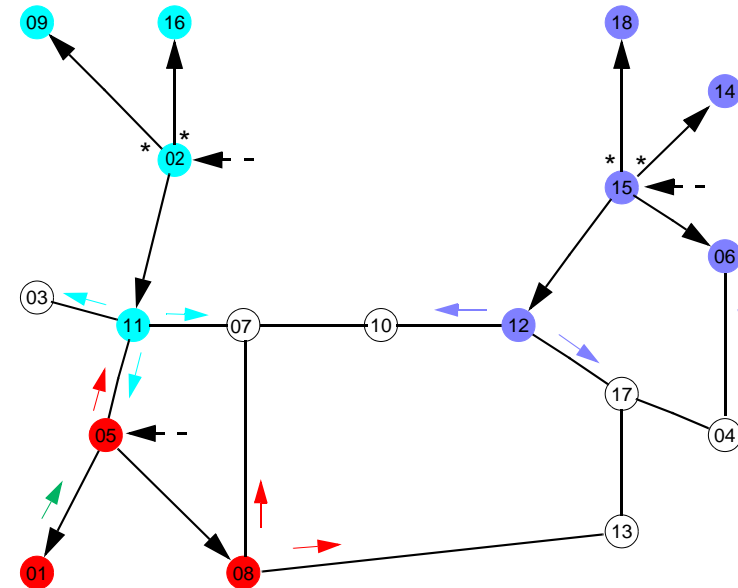
Wahl auf beliebigen Topologien



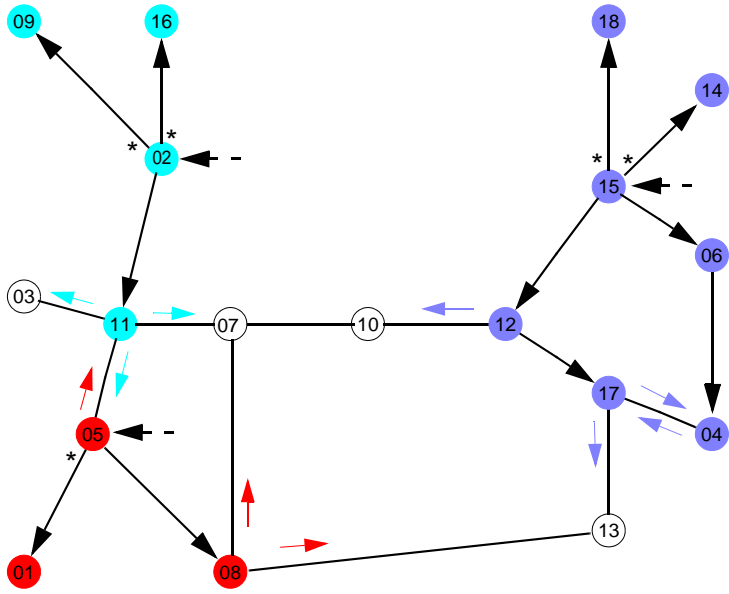
Wahl auf beliebigen Topologien



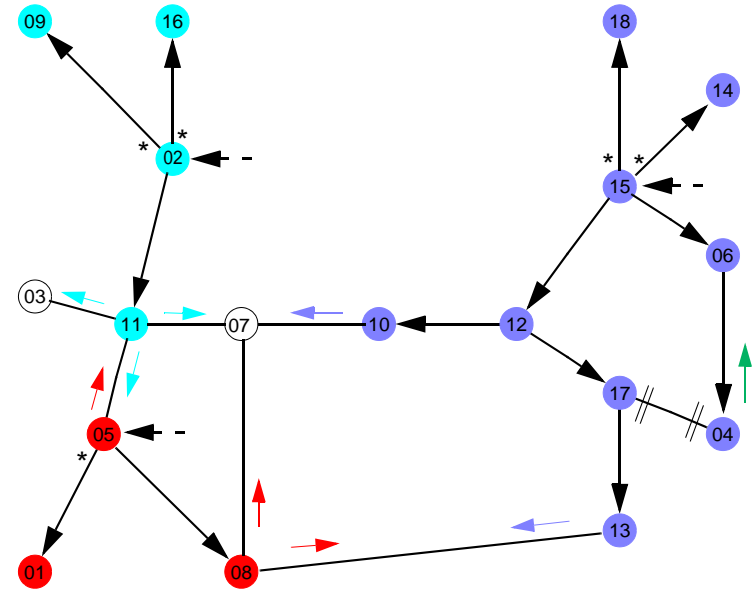
Wahl auf beliebigen Topologien



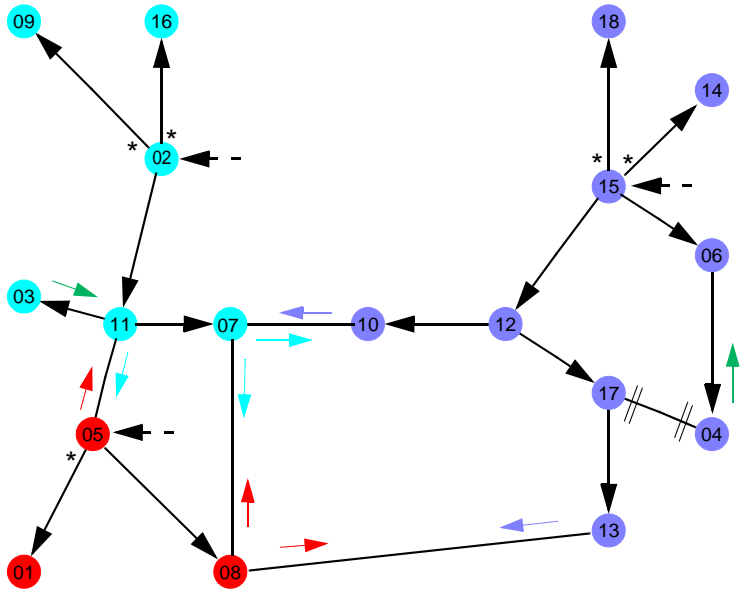
Wahl auf beliebigen Topologien



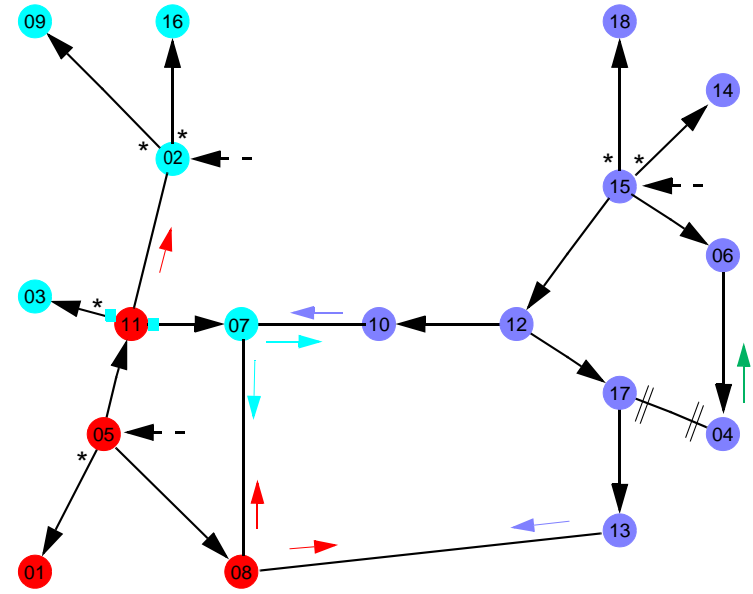
Wahl auf beliebigen Topologien



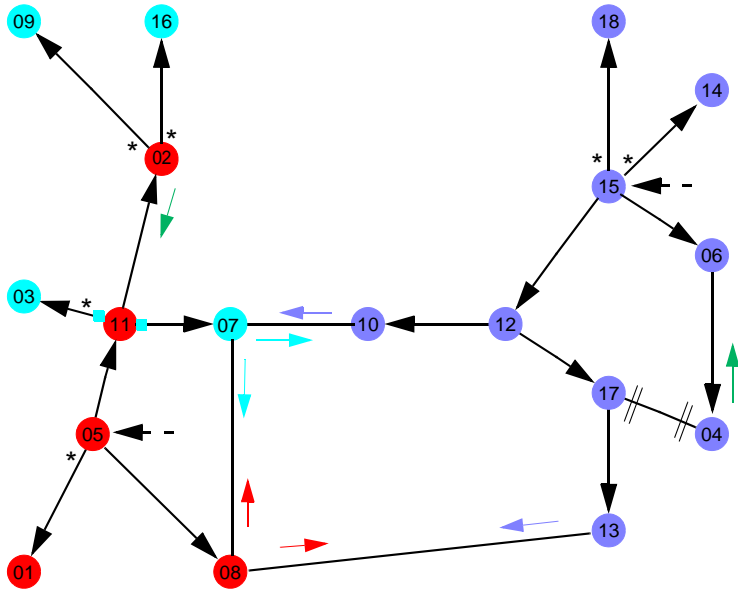
Wahl auf beliebigen Topologien



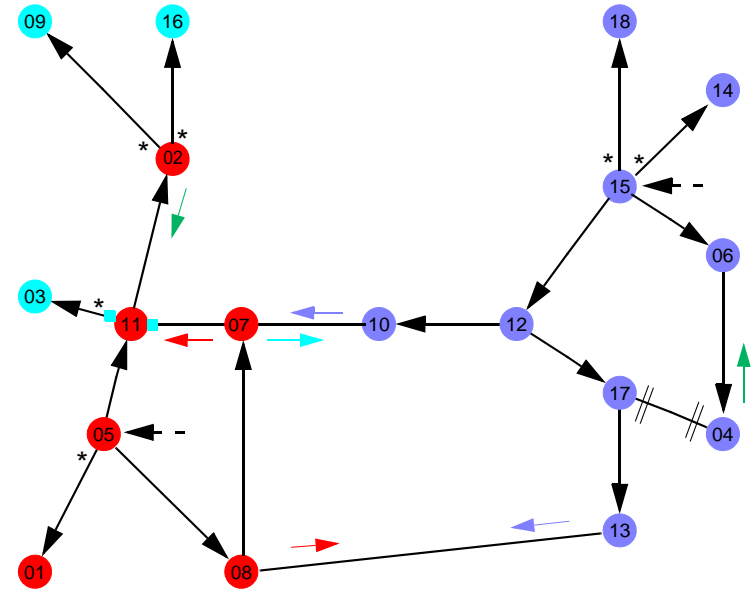
Wahl auf beliebigen Topologien



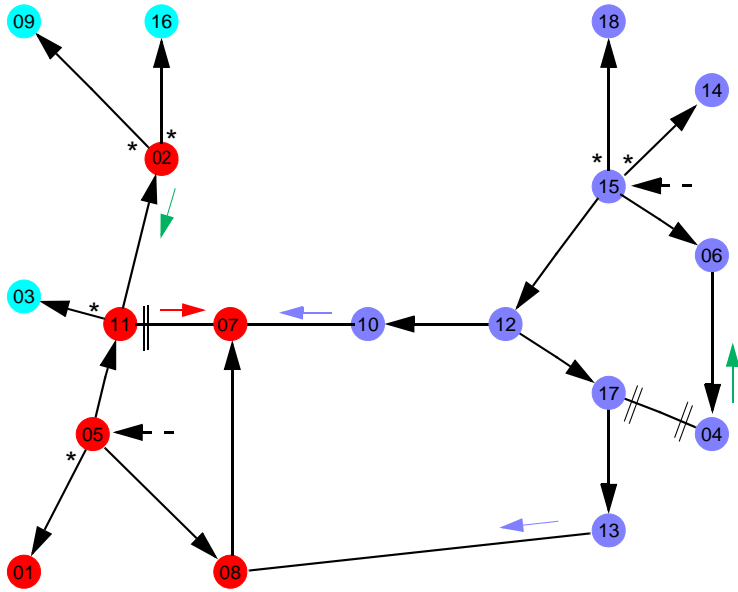
Wahl auf beliebigen Topologien



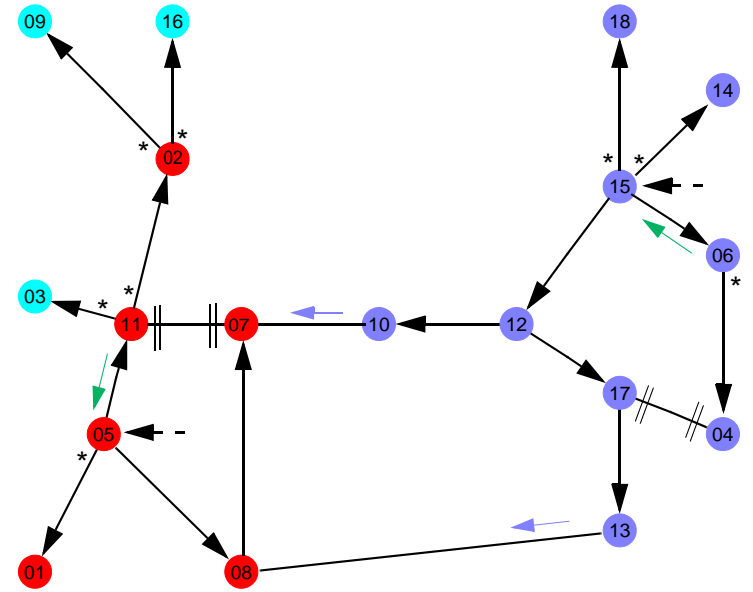
Wahl auf beliebigen Topologien



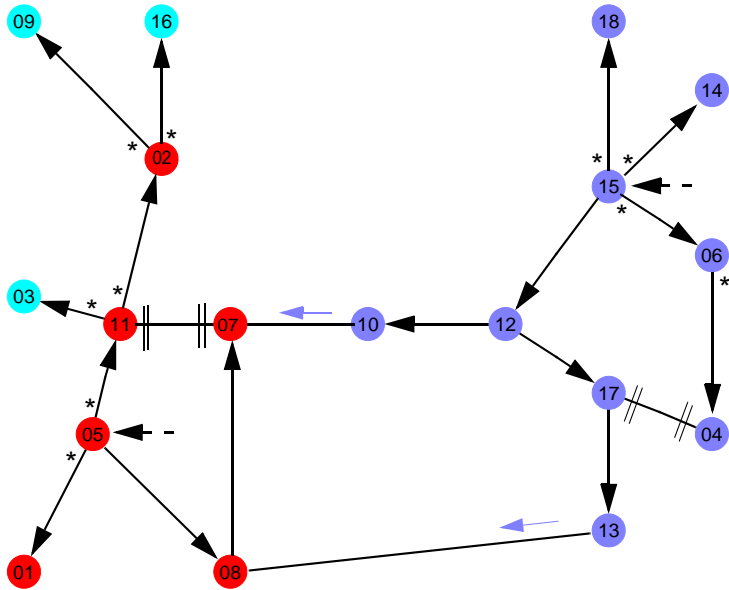
Wahl auf beliebigen Topologien



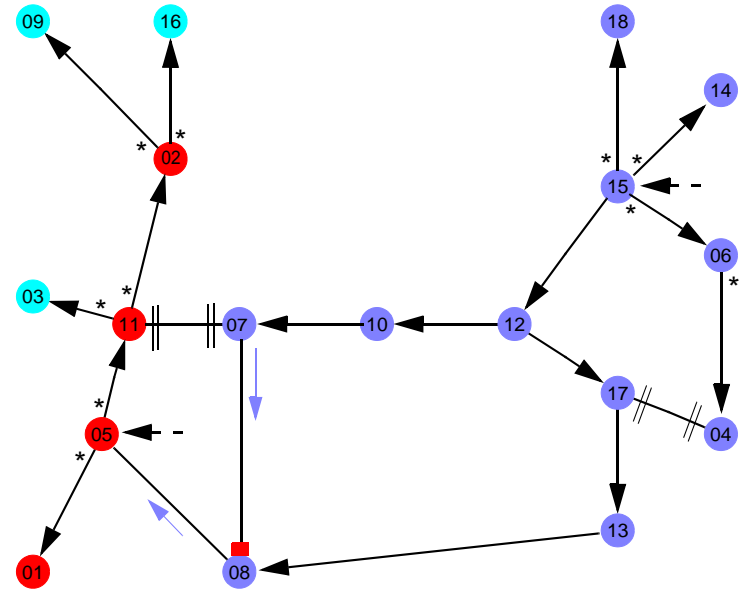
Wahl auf beliebigen Topologien



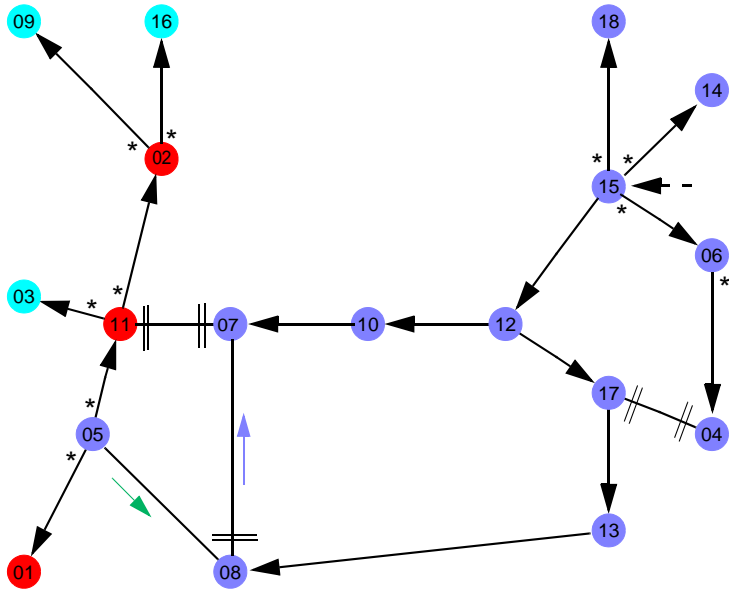
Wahl auf beliebigen Topologien



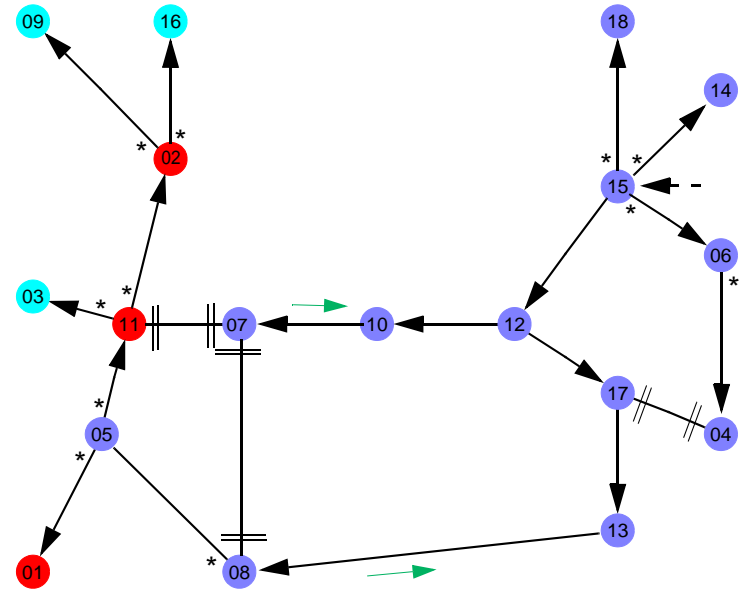
Wahl auf beliebigen Topologien



Wahl auf beliebigen Topologien



Wahl auf beliebigen Topologien



Wahl auf beliebigen Topologien

Zusammenfassung

- ▶ Wahlalgorithmen auf beliebigen Topologien: Algorithmen für Baumstrukturen + Zyklenerkennung
- ▶ Explorer- und Echowelle, ggf. Informationswelle
- ▶ Konstruktion eines virtuellen Baumes
- ▶ Verschiedene Varianten möglich:
 - ▶ Sequentielle Traversierung einfach, aber hoher zeitlicher Aufwand
 - ▶ Parallele Traversierung komplexer, aber wesentlich geringere Zeitkomplexität