Überblick

Motivation

- Problem: Wahl eines Anführerknotens
  - Beispielszenarien
    - Koordinierung verteilter Aktionen
    - Erzeugung systemweit eindeutiger Token
    - Passive Replikation
  - Anlässe
    - Initialisierung im Rahmen des Systemstarts
    - Neukonfigurierung als Reaktion auf Fehler
- Anforderungen
  - *Eindeutigkeit*: Zu jedem Zeitpunkt ist maximal ein Knoten der Anführer
  - *Terminierung*: Bestimmung des Anführers erfolgt in endlicher Zeit
- Beispiele für zusätzliche Kriterien
  - Deterministischer Wahlalgorithmus
  - Benachrichtigung aller Knoten über das Ende bzw. Ergebnis der Wahl

Motivation

- Systemmodell
  - Verteiltes System mit potenziell sehr vielen Knoten
  - Knoten
    - Gesamtzahl aller Knoten ist unbekannt
    - Nicht alle Knoten sind kontinuierlich Teil des Systems
    - Bedingung für deterministische Wahl: Jeder Knoten hat eindeutige ID
  - Unterschiedliche Netztopologien
    - Ring
    - Baum
    - Beliebige Strukturierung

  → Nicht jeder Knoten ist notwendigerweise mit jedem anderen verbunden
- Herausforderungen
  - Wahlalgorithmen für nicht vollvermaschte Netzwerke
  - Effizienzsteigerung durch Ausnutzen von Wissen über die Netztopologie
  - Anführerwahl in Systemen mit komplexer Netztopologie

Literatur

- Wahl auf Ringen


Nancy Lynch
Distributed Algorithms
Morgan Kaufmann Publishers Inc., 1996.


Gérard Le Lann
Distributed Systems – Towards a Formal Approach
Proceedings of the IFIP Congress 77, 7:155–160, 1977.


Ernest Chang and Rosemary Roberts
An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes
Communications of the ACM, 22(5):281–283, 1979.


Gary L. Peterson
An O(n log n) Unidirectional Distributed Algorithm for the Circular Extrema Problem
ACM Transactions on Programming Languages and Systems, 4(4):758–762, 1982.

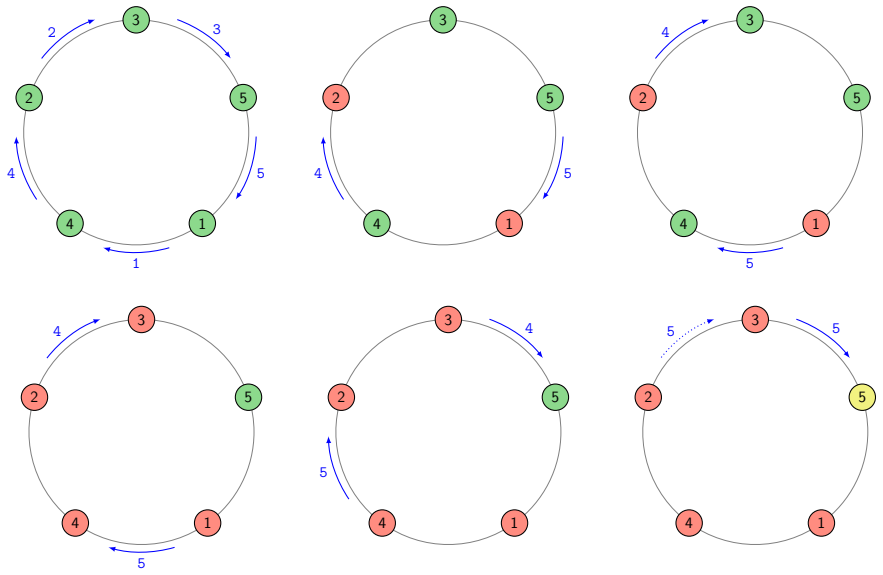
- Wahl auf Bäumen und beliebigen Graphen


Friedemann Mattern
Verteilte Basisalgorithmen
Springer-Verlag, 1989.

Chang-Roberts-Algorithmus — Übersicht

- Voraussetzungen
  - Jeder potenziell im System befindliche Knoten besitzt eine eindeutige ID
  - Auf den IDs ist eine totale Ordnung definiert (z. B. ID $\in [1,\ldots,n]$)
- Ziel: Bestimmung des aktiven Knotens mit der höchsten ID
- Ablauf
  - Start des Wahlalgorithmus
    - Annahme: Alle Knoten starten Algorithmus zur selben Zeit
    - Jeder Knoten sendet eigene Knoten-ID im Ring weiter
    - Jeder Knoten wird wählbar
  - Empfang einer ID i
    - Falls $i >$ eigene ID — Knoten sendet i weiter; Knoten ist nicht mehr wählbar
    - Falls $i <$ eigene ID — Nachricht ignorieren
    - Falls $i =$ eigene ID — Falls Knoten wählbar, dann ist er als Anführer gewählt; Sonst: Nachricht ignorieren

Chang-Roberts-Algorithmus — Beispiel



Chang-Roberts-Algorithmus — Analyse

- Bemerkungen
  - Einfacher Algorithmus auf Basis eines einfachen Systemmodells
  - Teilnehmer müssen die Gesamtzahl aller Knoten n nicht kennen
- Mögliche Erweiterung
  - Anführer sendet nach seiner Wahl eine Benachrichtigung durch den Ring
  - Sauberes Beenden des Wahlalgorithmus auf allen Knoten möglich
- Korrektheit
  - Eindeutigkeit
    - Nachricht mit einer bestimmten ID wird von genau einem Knoten erzeugt
    - Knoten mit höchster ID leitet nur seine eigene Nachricht weiter
    - Ein Knoten wird nur gewählt, wenn seine ID von allen weitergereicht wurde
  - Terminierung
    - Die Nachricht mit der größten ID wird von jedem Knoten weitergereicht
    - Nach n Schritten kommt die größte ID wieder beim Ursprungsknoten an
    - Achtung: Terminierung ist bei Ausfällen von Knoten nicht sichergestellt

Chang-Roberts-Algorithmus — Komplexität

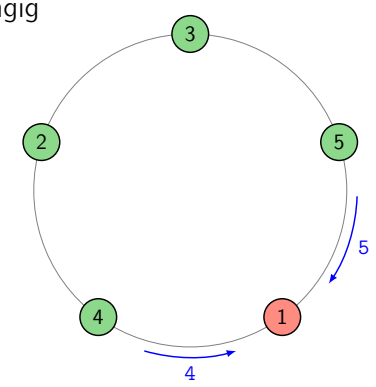
- Annahmen bei der Analyse der Zeitkomplexität
  - Bearbeitungsdauer von Nachrichten ist vernachlässigbar
  - Nachrichten kommen innerhalb einer maximalen Nachrichtenlaufzeit an
- Bester Fall: Knoten sind den IDs nach aufsteigend angeordnet
  - n Nachrichtenlaufzeiten
  - $n + (n-1) \cdot 1 = 2n - 1$ Nachrichten $\rightarrow O(n)$
- Schlimmster Fall: Knoten sind den IDs nach absteigend angeordnet
  - n Nachrichtenlaufzeiten
  - $\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$ Nachrichten $\rightarrow O(n^2)$
- Durchschnittlicher Fall
  - Annahme: Alle $(n-1)!$ möglichen ID-Verteilungen gleichwahrscheinlich
  - n Nachrichtenlaufzeiten
  - Nachrichtenaufwand: $O(n \log n)$ (siehe [Chang et al.])

Chang-Roberts-Algorithmus — Initiierung

- Veränderte Startbedingung
  - Gleichzeitiger Start des Algorithmus auf allen Knoten unrealistisch
  - Ein Knoten startet den Algorithmus durch Senden seiner ID
  - Andere Knoten beteiligen sich erst, nachdem sie eine Nachricht erhalten
- Zeitkomplexität
  - Bester Fall
    - Knoten mit höchster ID initiiert Wahl
    - n Nachrichtenlaufzeiten
  - Schlechtester Fall
    - Knoten mit höchster ID wacht als Letzter auf
    - $(n-1)+n = 2n-1$ Nachrichtenlaufzeiten
- Optimierung
  - Unterdrückung der eigenen Nachricht, falls Aufwecken durch größere ID
  - Reduzierung der im besten Fall erforderlichen Nachrichten auf n

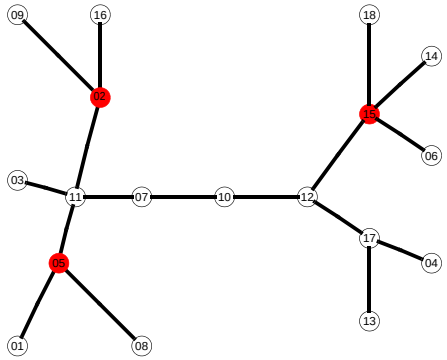
Chang-Roberts-Algorithmus — Variante

- Veränderte Architektur: Bidirektionaler Ring
- Unterschiede zur Standardvariante
  - Start: Jeder Knoten bestimmt die Senderichtung der eigenen ID zufällig
  - Jeder Knoten speichert die höchste ID i_{max}, die er bisher gesehen hat
  - Unterdrückung einer neu empfangenen ID i, falls $i < i_{max}$
  - Senderichtung der Weiterleitung abhängig von bisheriger Senderichtung einer ID
- Bemerkungen
  - Geringere Anzahl erforderlicher Nachrichten im Durchschnittsfall
  - Schlimmster Fall
    - Geringere Auftrittswahrscheinlickeit
    - Nachrichtenaufwand weiterhin $O(n^2)$



Wahl auf Bäumen — Übersicht

- Baumstruktur
  - Ungerichtete Kanten
  - Keine Zyklen
- Vergleich zur Wahl auf Ringen
  - Kommunikation mit unterschiedlich vielen Nachbarknoten
  - Potenzial für *Divide-and-Conquer*-Ansätze



Wahl auf Bäumen — Traversieren

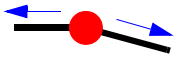
- Sequentielles Traversieren – Beispiel: Tiefensuche

```
public class VSNode {
  private final int id = [ID des lokalen Knotens];
  private final List<VSNode> children = [Liste der Kindknoten];

  public int findMaxID() {
    int maxID = id;
    for(VSNode child: children) {
      int childMaxID = child.findMaxID();
      if(maxID < childMaxID) maxID = childMaxID;
    }
    return maxID;
} }
```

- Paralleles Traversieren – Beispiel: Wellenverfahren
  - *Explorationswelle* durchläuft den Baum bis zu den Blättern
  - *Echowelle* kommt zurück und bestimmt die höchste ID
  - *Informationswelle* teilt allen Knoten das Ergebnis mit

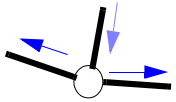
Wahl auf Bäumen — Algorithmus

- Dedizierte Initiatorknoten senden Explorernachrichten an alle ihre Nachbarknoten

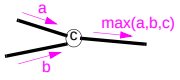


- Blattknoten senden bei Empfang einer Explorernachricht die eigene ID als Echonachricht zurück

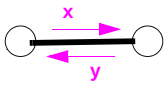
- Innere Knoten mit k Kanten
  - Weiterleitung der ersten Explorernachricht in alle übrigen Richtungen



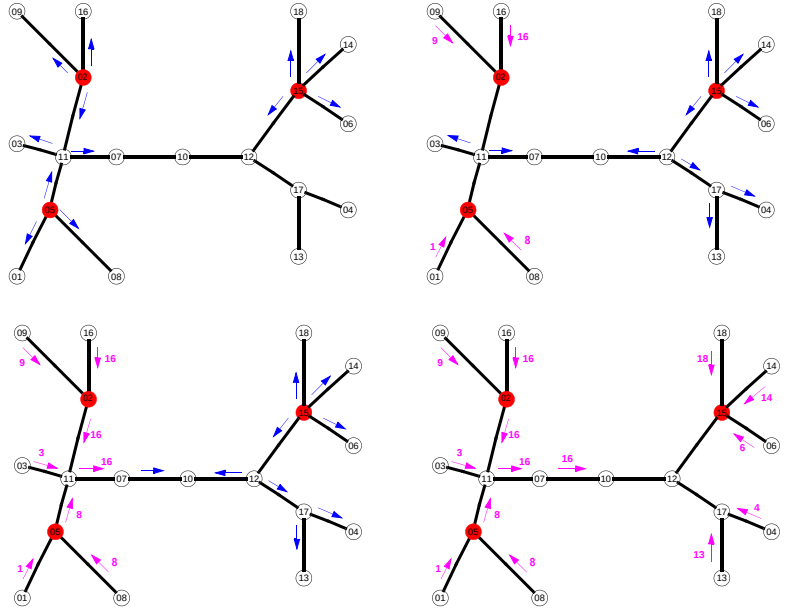
  - Nach Erhalt von Echonachrichten auf $k-1$ Kanten: Senden einer Echonachricht e mit dem Maximum aller bisher bekannten IDs über die verbleibende Kante



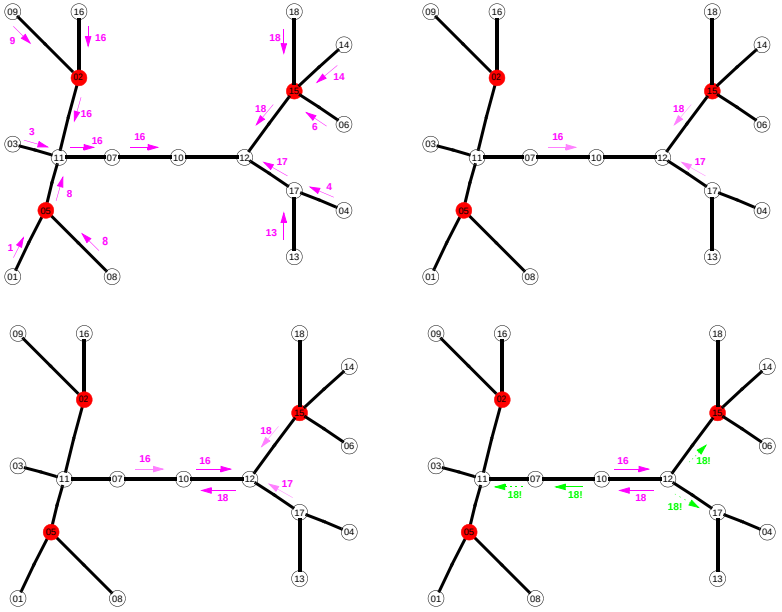
  - Falls Empfang einer weiteren Echonachricht e': Höchste ID im Netz ist Maximum der IDs von e und e'



Wahl auf Bäumen — Beispiel



Wahl auf Bäumen — Beispiel



Wahl auf beliebigen Topologien — Überblick

- Annahmen über Netzstruktur
  - Zusammenhängender Netzwerkgraph
  - Bidirektionale Verbindungen zwischen Knoten
  - Netzwerkgraph kann Zyklen aufweisen
- Sequentielles Traversieren – Beispiel: Tiefensuche
  - Initiatorknoten erstellt Token und sendet es über eine Kante k_{init}
  - Empfang des Tokens über eine Kante k_{recv}
    - Weitergabe des Tokens an einen Nachbarn, der das Token noch nicht hatte
    - Falls kein solcher Nachbar bekannt: Senden des Tokens über Kante k_{recv}
  - Zyklenerkennung anhand der Kante, über die Token empfangen wurde
  - Terminierung, sobald Initiator das Token über die Kante k_{init} empfängt
- Alternativer Ansatz: Konstruktion eines virtuellen Baums
  - Initial beim Systemstart oder
  - Im Nachhinein mittels Adoptionsverfahren

Erzeugung eines virtuellen Baums beim Systemstart

- Anmeldung bei einer globalen Registry
  - Registry kennt die Adressen aller bereits im System befindlicher Knoten
  - Bereitstellung von Knotenadressen für Verbindungsaufbau
- Verbindungsaufbau
  - Verwaltung eines lokalen Levels
  - Wurzelknoten des Baums hat Level 0
  - Elternknoten: Nachbar mit kürzester Distanz zum Wurzelknoten
- Wiederholung der Auswahl bei Abbruch der Elternverbindung

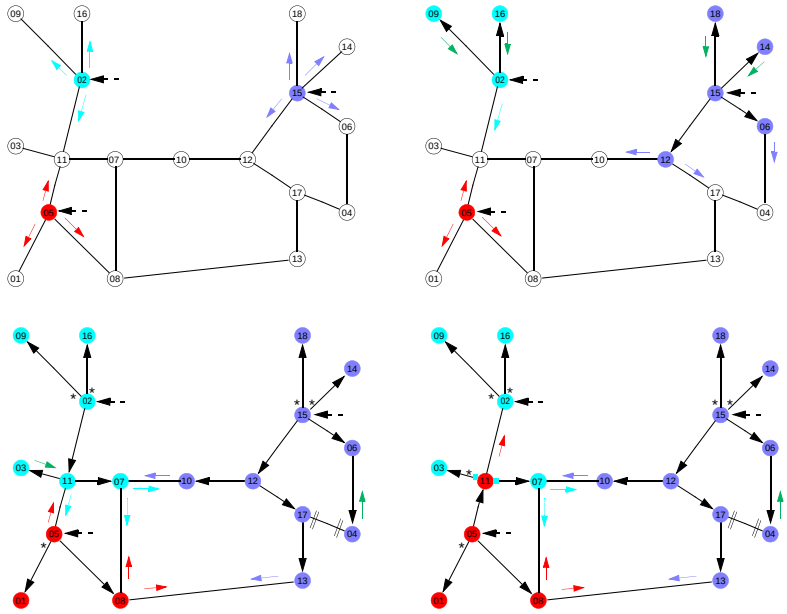
```
private int level = -1;
private VSNode parent = null;

public void connect(List<VSNode> nodes) {
  for(VSNode node: nodes) {
    // Verbindungsaufbau
    boolean connected = [Aufbau der Verbindung];
    if(!connected) continue;

    // Bestimmung des Elternknotens
    if((level < 0) || (level > node.level)) {
      level = node.level + 1;
      parent = node;
} } }
```


Adoptionsverfahren — Algorithmus

- Grundprinzip wie bei Wahl auf Bäumen: Explorations- und Echowelle
- Nachrichten
  - Explorernachrichten beinhalten ID des korrespondierenden Initiators
  - Echonachrichten tragen Initiator-ID der zugehörigen Explorernachricht
- Verwaltung von IDs an Knoten und Kanten
  - Speicherung der höchsten einem Knoten bekannten Initiator-ID (ID_i)
  - *Elternkante*: Kante, über die Explorernachricht mit ID_i empfangen wurde
  - Markierung der Sendekante einer Explorernachricht mit Initiator-ID
- Empfang einer Explorernachricht x über Kante k mit Markierung m

Falls $ID_x = ID_m$	Kante k ist nicht Teil des virtuellen Baums
Falls $ID_x < ID_m$	Ignorieren der eintreffenden Explorernachricht x
Falls $ID_m < ID_x \leq ID_i$	Aktualisierung von m, Senden von x über k
Falls $ID_m \leq ID_i < ID_x$	*Adoption*: k wird neue Elternkante Weiterleitung von x über bisherige Elternkante

Adoptionsverfahren — Beispiel



Adoptionsverfahren — Beispiel