Kapitel 10
Parallele Algorithmen

- Bitonisches Sortieren
- Teilworterkennung
- Leader election in Ringnetzwerken

Parallele Algorithmen
Überblick

- Situation:
  - weitere Beschleunigung der Prozessorgeschwindigkeit wünschenswert
  - physikalischer Grenzwert "Lichtgeschwindigkeit" bald erreicht
  - Offenbar einzige Lösung: Parallelisierung

Parallele Algorithmen
Überblick

- Wunsch:
  - p parallel arbeitende Prozessoren lösen ein Problem p-fach schneller (10^5 bis 10^6 parallele Prozessoren werden angestrebt)
- Problem
  - Geeignete Parallelisierung eines Problems erfordert neue Algorithmen

Parallele Algorithmen
Beispiel

- Addition von Vektoren $a=(a_1,...,a_n)$ und $b=(b_1,...,b_n)$ leicht parallelisierbar
- Berechnung der Potenz x^{2^k} beweisbar nicht effizient parallelisierbar

Parallele Algorithmen
Sortieren

- Eingabe: n ganze Zahlen $a_1,...,a_n$ (o.B.d.A. n Zweierpotenz, sonst ergänze zur nächsten Zweierpotenz um Zahlen ∞)
- Ausgabe: aufsteigende Sortierung von $a_1,...,a_n$
- Algorithmus: Bitonisches Sortieren (K.E. Batcher, 1968)

Parallele Algorithmen
bitonisch

Definition:

$a_1,...,a_n$ heißt **bitonisch**, falls $a_1,...,a_i$ aufsteigend und $a_{i+1},...,a_n$ absteigend sortiert ist, oder falls dieser Zustand durch *Rundschieben* hergestellt werden kann.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
bitonisch

Beispiele:

7, 12, 13, 16, 29, 17, 12, 1

16, 29, 17, 12, 1, 7, 12, 13

1, 2, 3, 4, 5, 6, 7, 8.

Parallele Algorithmen
Idee des Algorithmus

- Fundamentale Ideen: Rekursion und Divide-and-conquer
- Ist $a_1,..., a_n$ bereits bitonisch, so mische die beiden Teilfolgen zu einer sortierten Folge: Fertig.
- Anderenfalls gehe rekursiv vor und versuche den gewünschten Zustand herzustellen, indem die linke Hälfte aufsteigend, die rechte Hälfte absteigend sortiert wird.

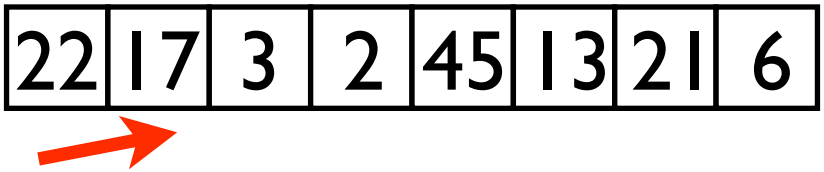
Parallele Algorithmen

Idee des Algorithmus - Beispiel

22	17	3	2	45	13	21	6

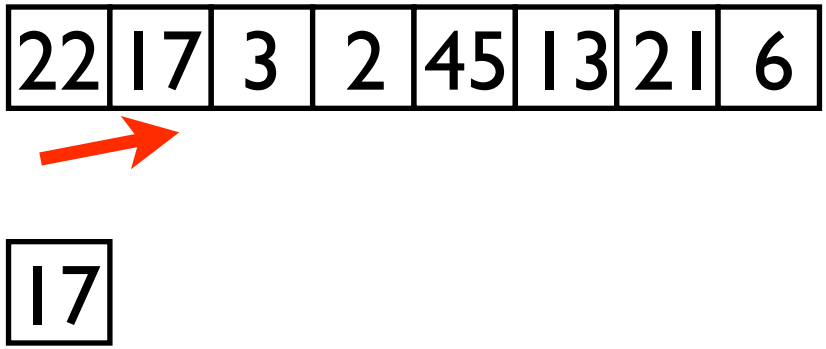
Parallele Algorithmen

Idee des Algorithmus - Beispiel



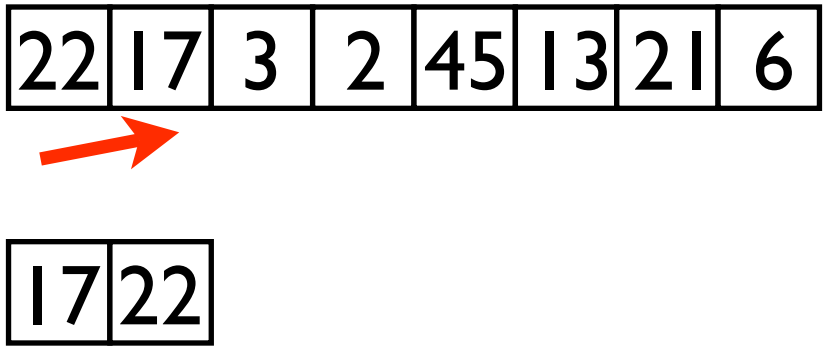
Parallele Algorithmen

Idee des Algorithmus - Beispiel



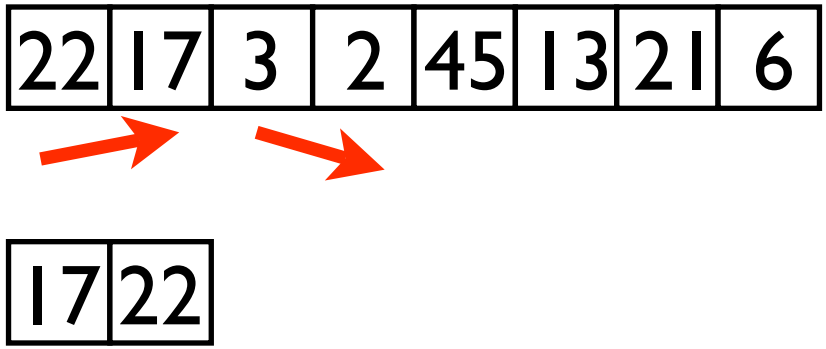
Parallele Algorithmen

Idee des Algorithmus - Beispiel



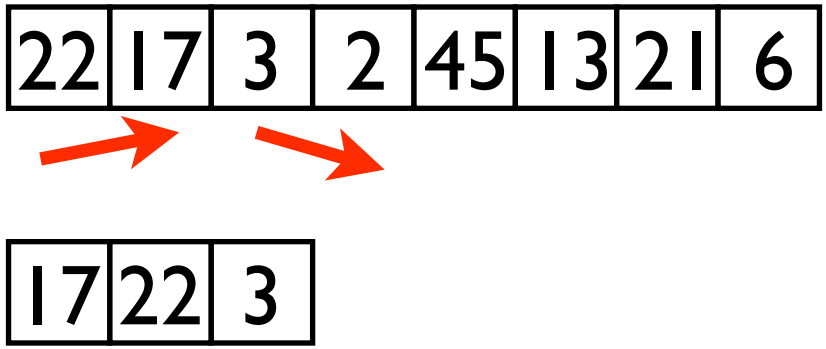
Parallele Algorithmen

Idee des Algorithmus - Beispiel



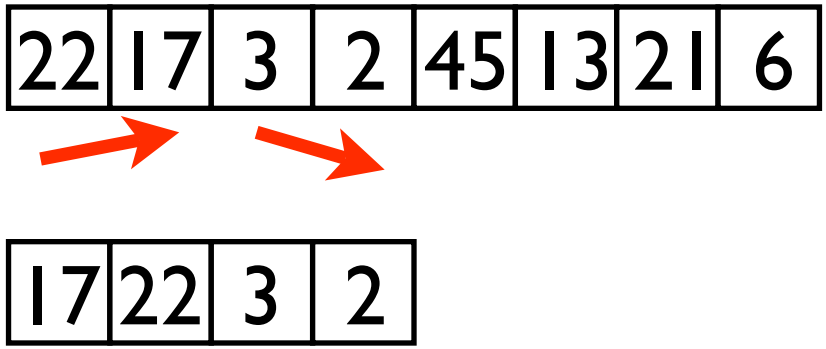
Parallele Algorithmen

Idee des Algorithmus - Beispiel



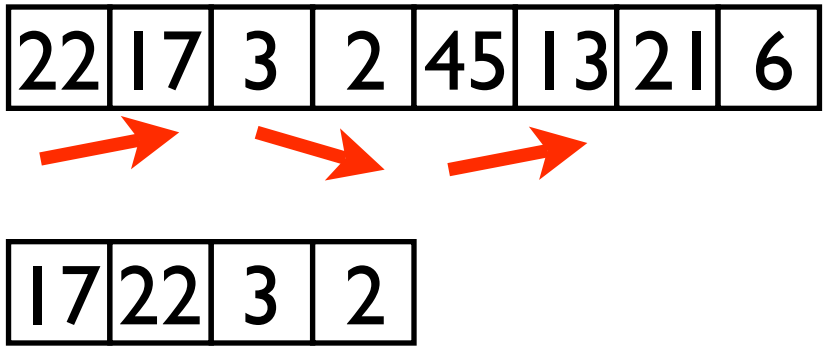
Parallele Algorithmen

Idee des Algorithmus - Beispiel



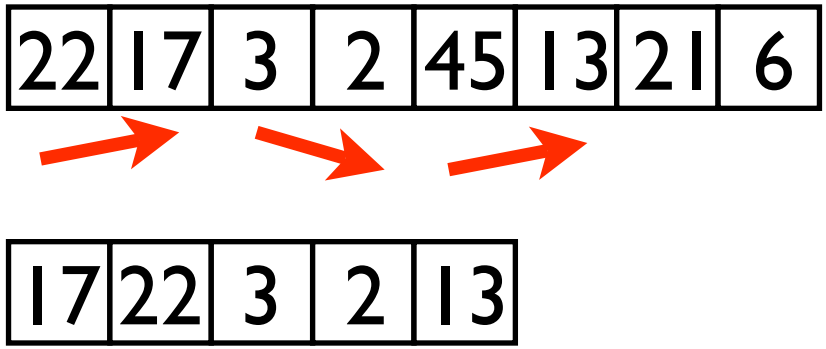
Parallele Algorithmen

Idee des Algorithmus - Beispiel



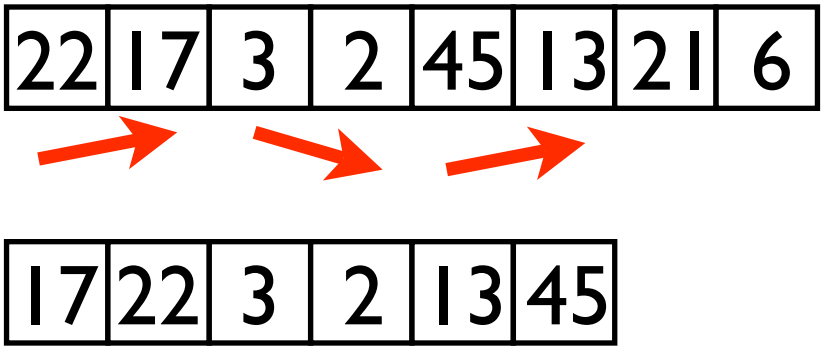
Parallele Algorithmen

Idee des Algorithmus - Beispiel



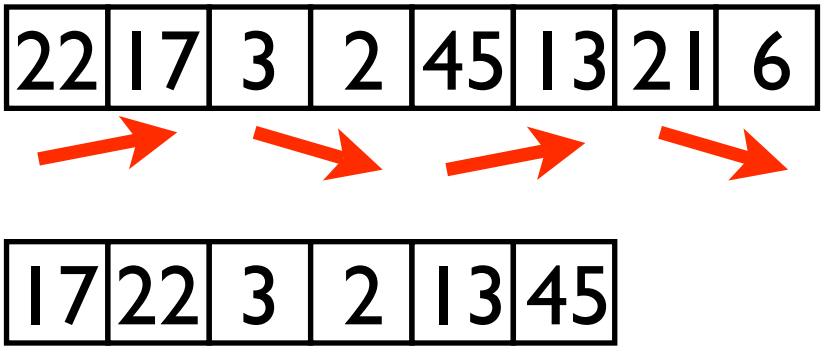
Parallele Algorithmen

Idee des Algorithmus - Beispiel



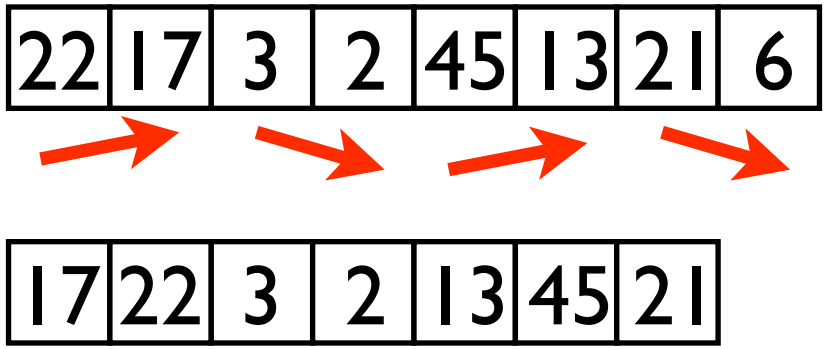
Parallele Algorithmen

Idee des Algorithmus - Beispiel



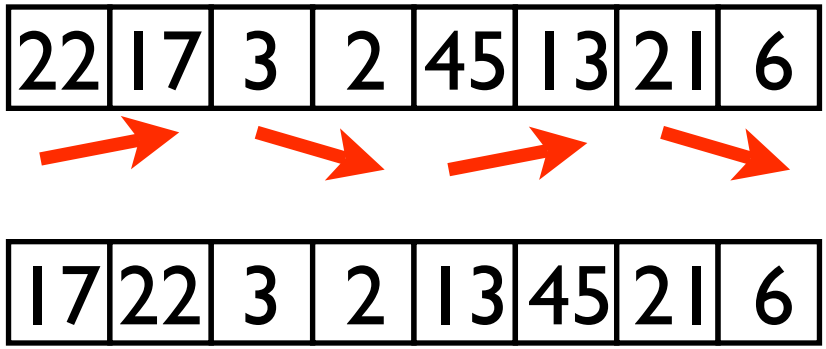
Parallele Algorithmen

Idee des Algorithmus - Beispiel



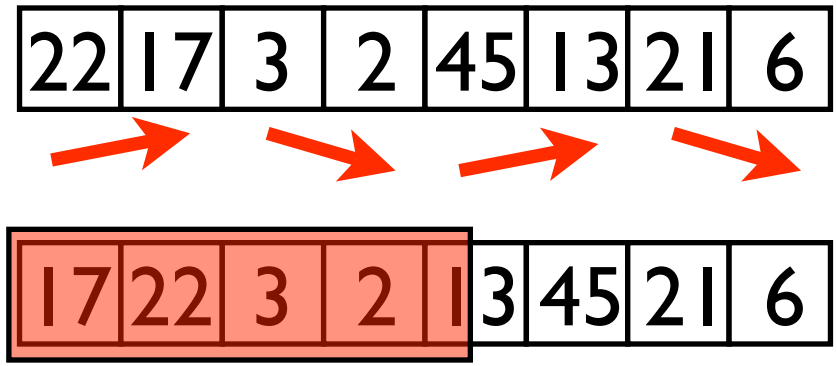
Parallele Algorithmen

Idee des Algorithmus - Beispiel



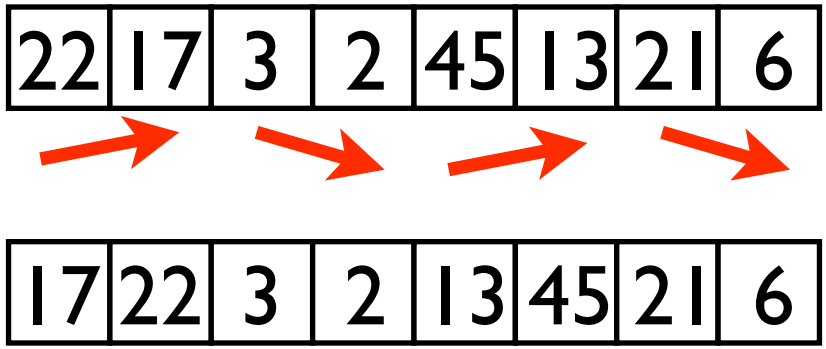
Parallele Algorithmen

Idee des Algorithmus - Beispiel



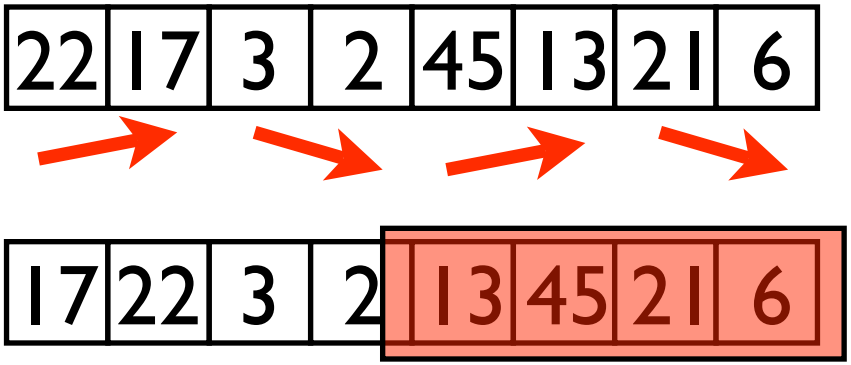
Parallele Algorithmen

Idee des Algorithmus - Beispiel



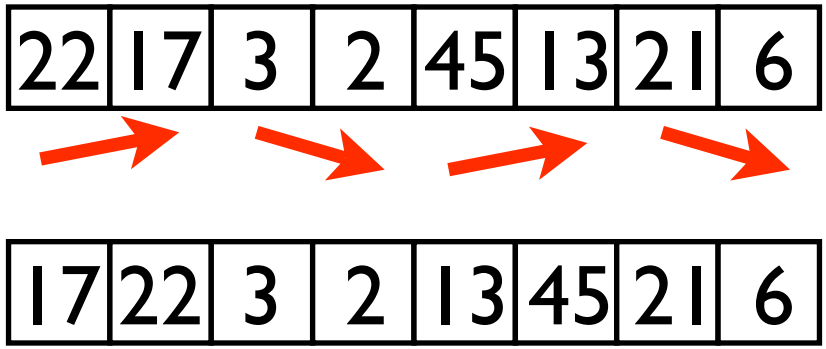
Parallele Algorithmen

Idee des Algorithmus - Beispiel



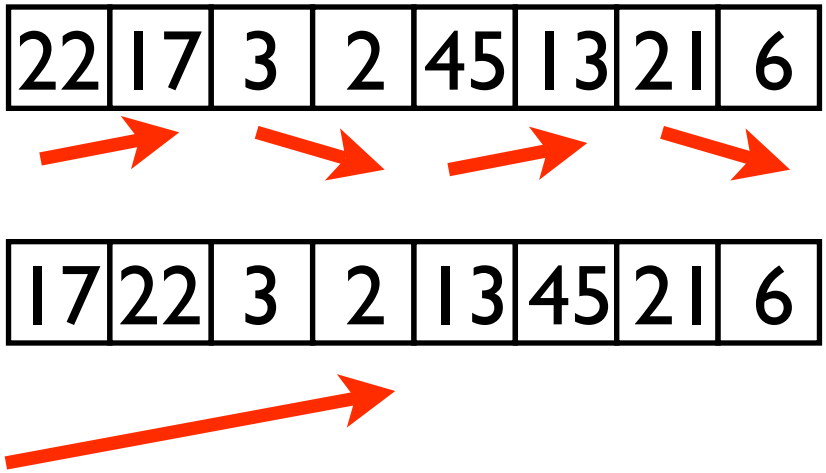
Parallele Algorithmen

Idee des Algorithmus - Beispiel



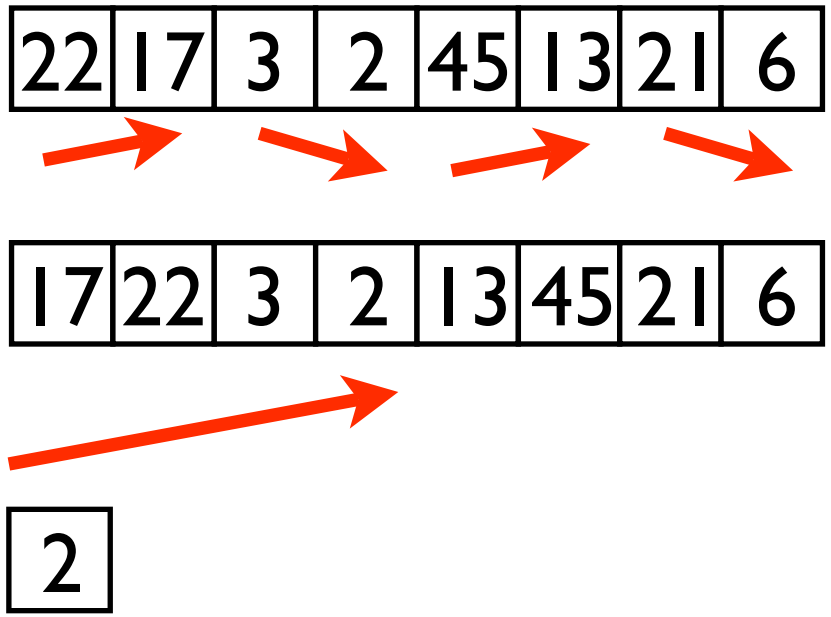
Parallele Algorithmen

Idee des Algorithmus - Beispiel



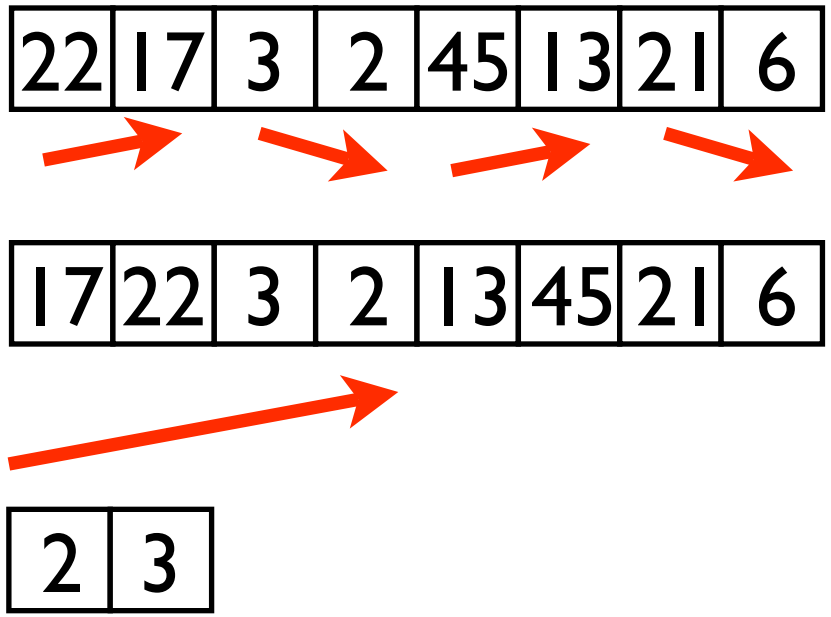
Parallele Algorithmen

Idee des Algorithmus - Beispiel



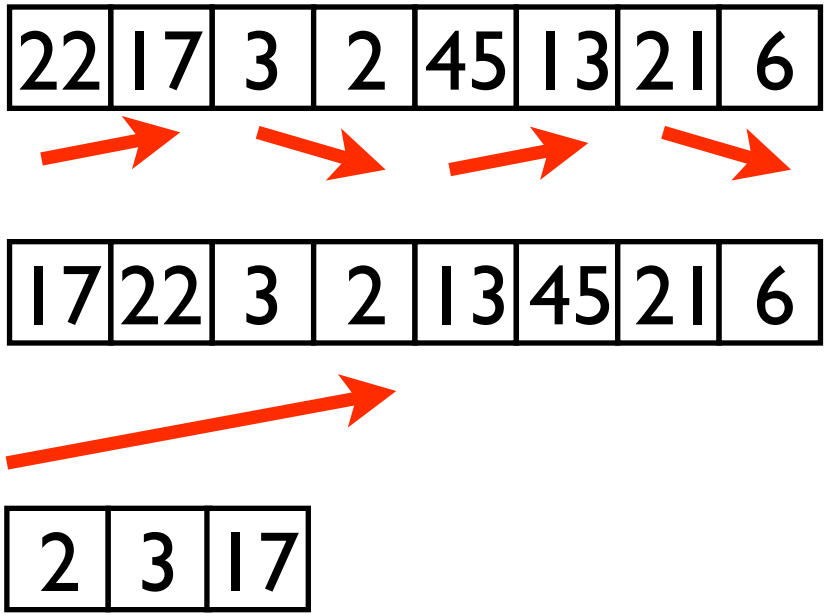
Parallele Algorithmen

Idee des Algorithmus - Beispiel



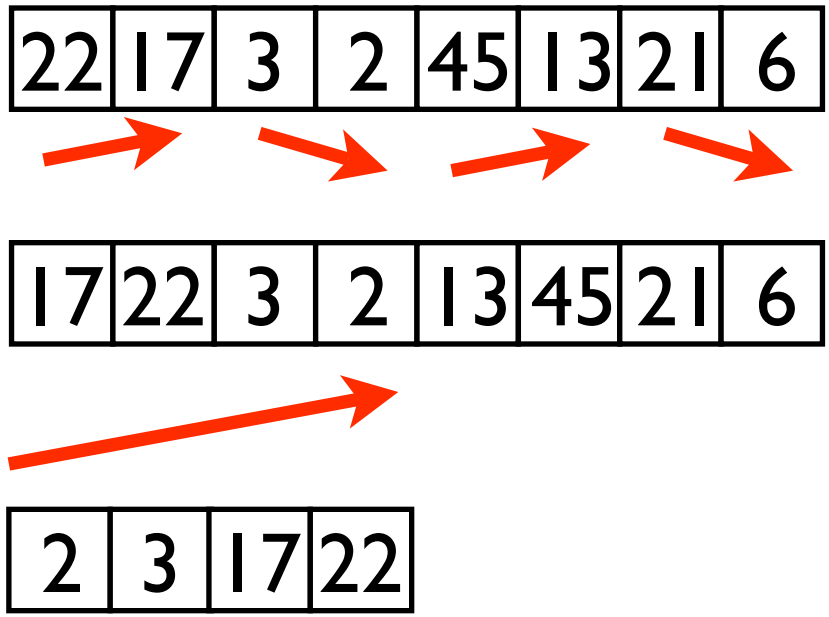
Parallele Algorithmen

Idee des Algorithmus - Beispiel



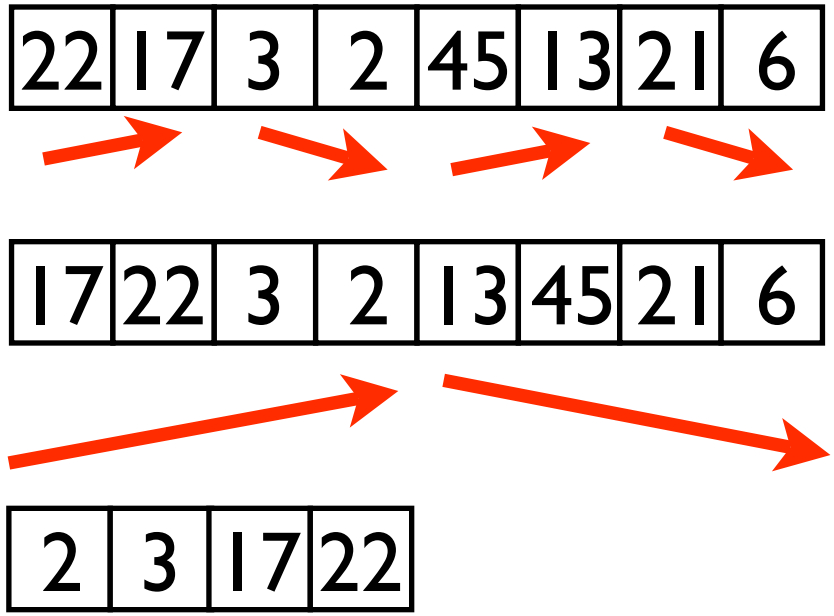
Parallele Algorithmen

Idee des Algorithmus - Beispiel



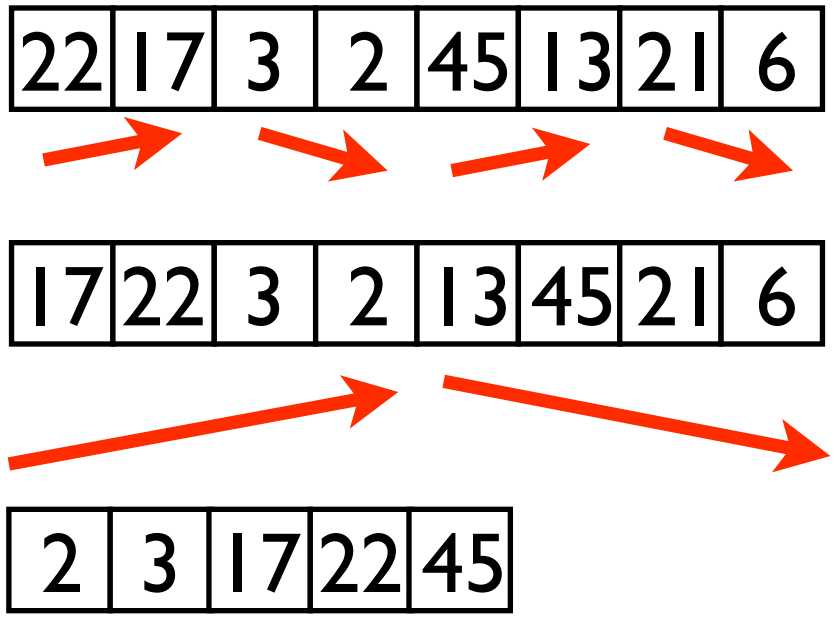
Parallele Algorithmen

Idee des Algorithmus - Beispiel



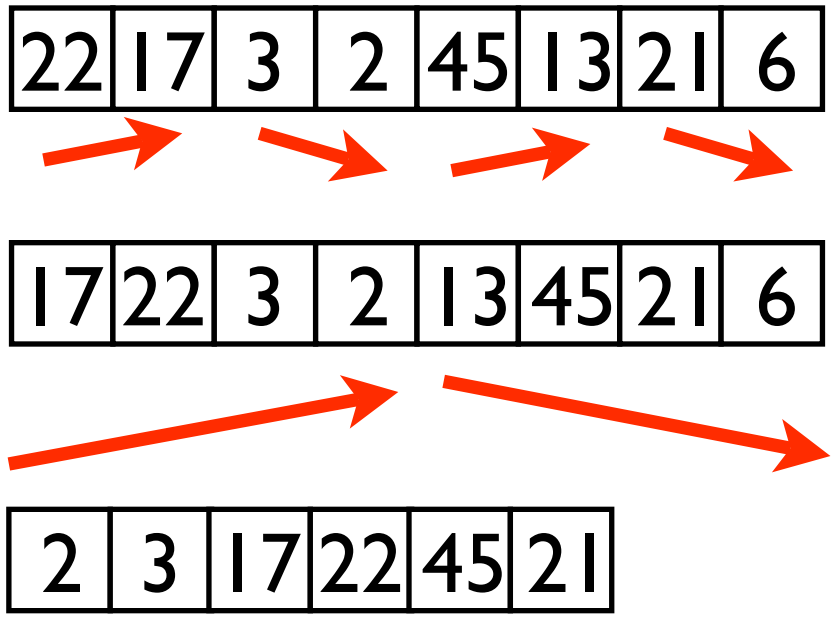
Parallele Algorithmen

Idee des Algorithmus - Beispiel



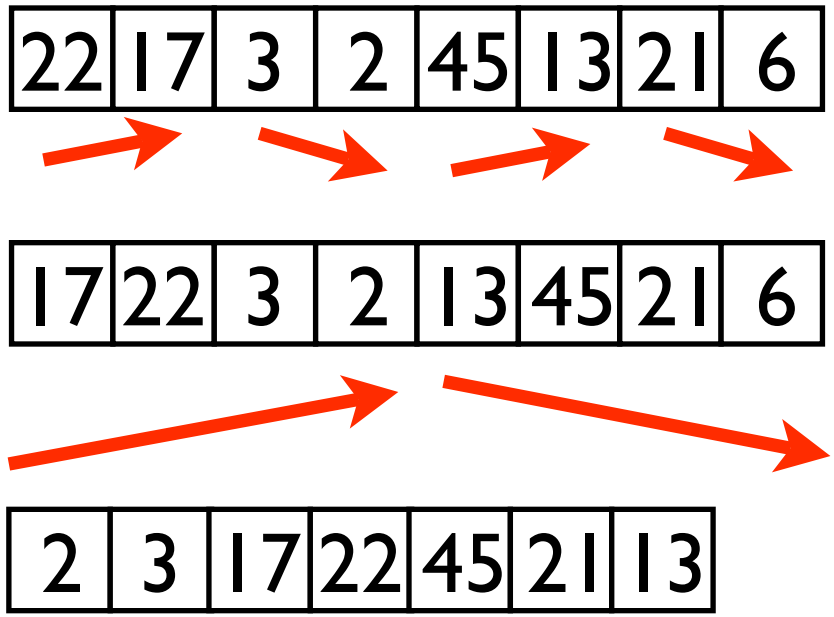
Parallele Algorithmen

Idee des Algorithmus - Beispiel



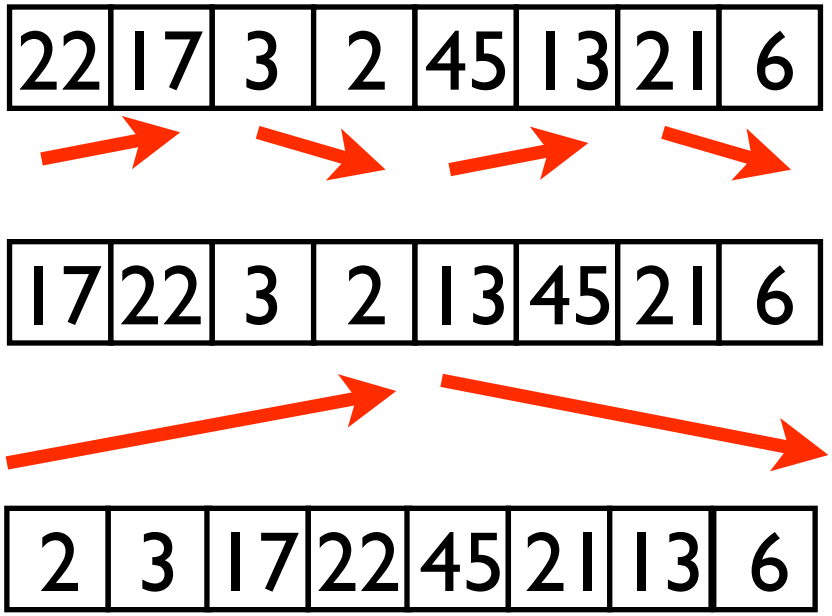
Parallele Algorithmen

Idee des Algorithmus - Beispiel



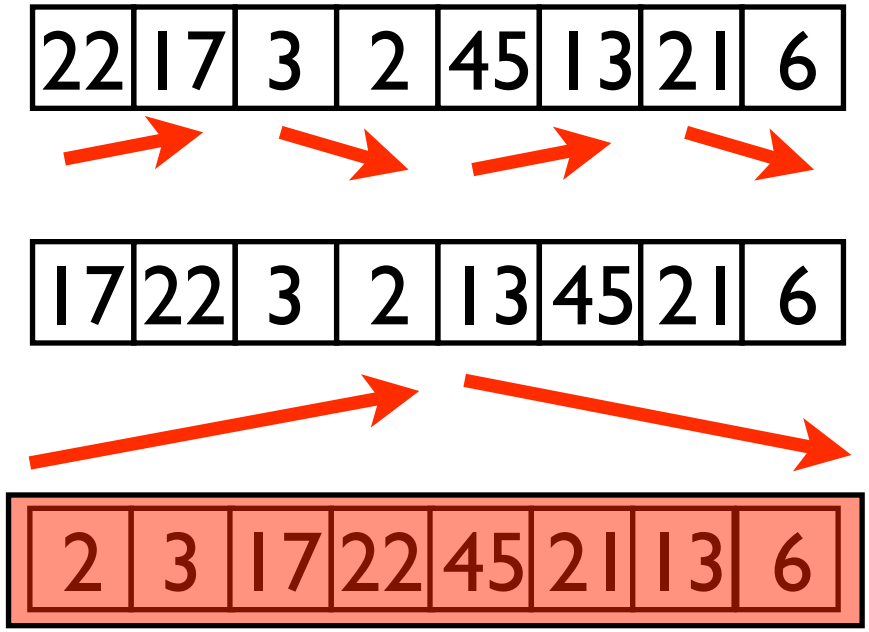
Parallele Algorithmen

Idee des Algorithmus - Beispiel



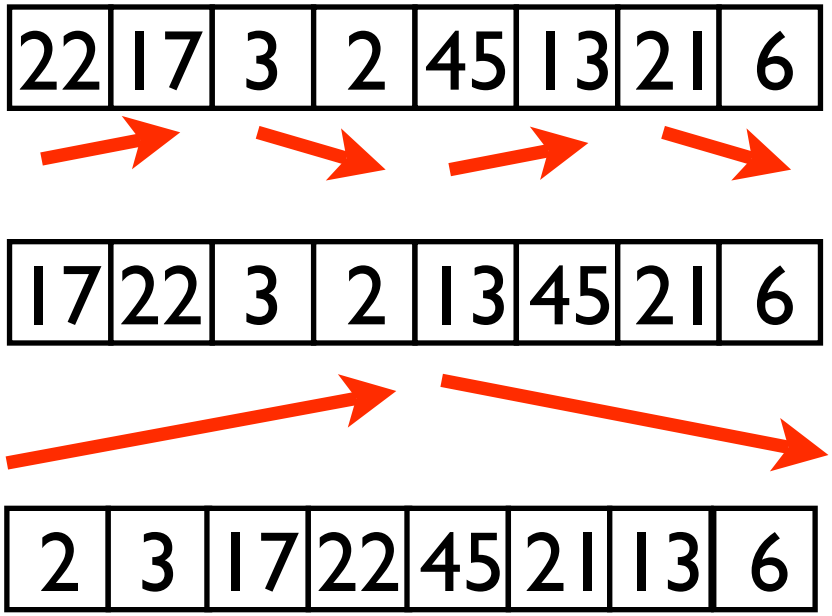
Parallele Algorithmen

Idee des Algorithmus - Beispiel



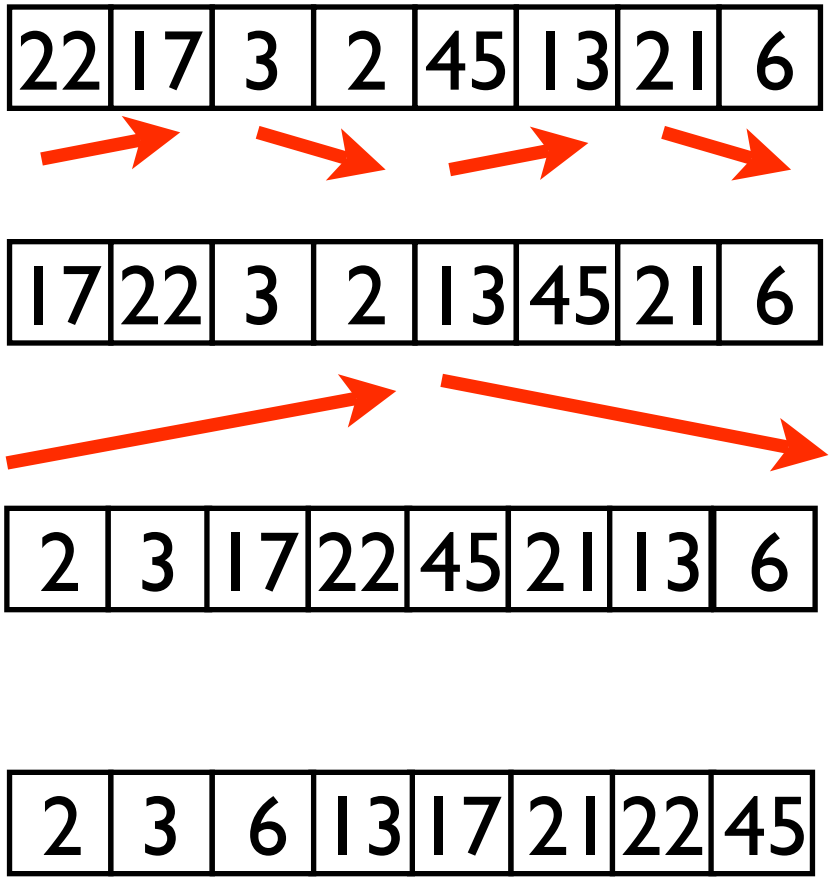
Parallele Algorithmen

Idee des Algorithmus - Beispiel



Parallele Algorithmen

Idee des Algorithmus - Beispiel



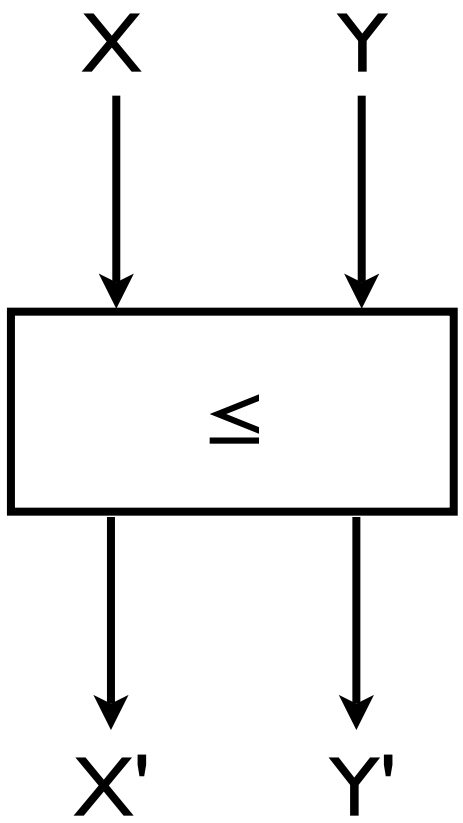
Parallele Algorithmen
Mischen einer bitonischen Folge

- Gegeben $a_1, a_2, \ldots, a_{2n}$
- 1. Schritt: Vergleiche jedes a_j, $1 \leq j \leq n$, der ersten Hälfte mit korrespondierendem Element a_{j+n} der zweiten Hälfte; vertausche, wenn sie in der falschen Reihenfolge stehen, falls also $a_j > a_{j+n}$ ist
- 2. Schritt: Wende Verfahren rekursiv auf die beiden Hälften an
- Fertig: Folge sortiert, wenn Verfahren abbricht

Parallele Algorithmen

Prozessor

$X' := \min(X,Y)$
$Y' := \max(X,Y)$
$X,Y,X',Y' \in \mathbb{N}$



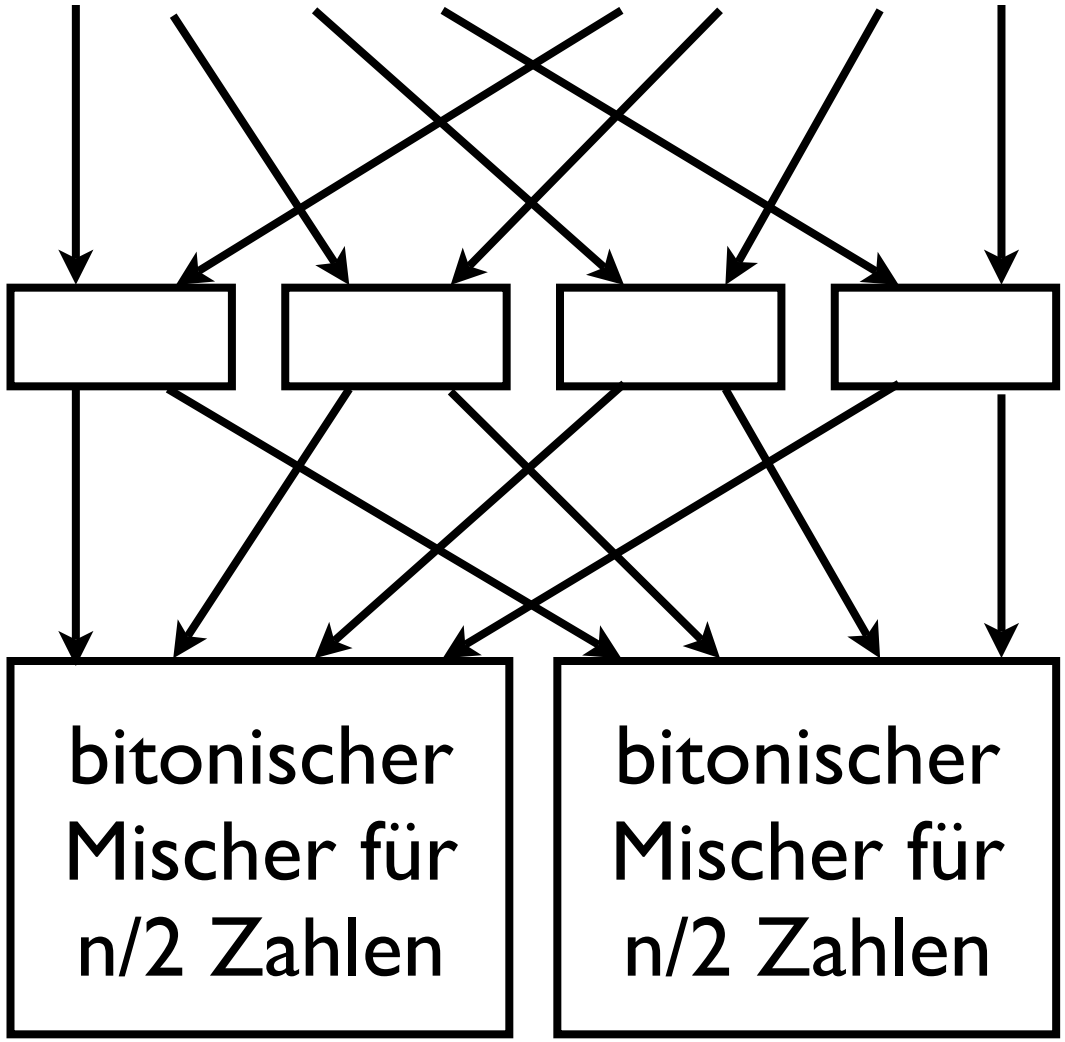
Parallele Algorithmen
Prozessornetzwerk - Grundoperationen

Grundoperationen:

- Zusammenführen aller Paare a_j und a_{j+n}, $1 \leq j \leq n$,
- Vergleichen
- ggf. Vertauschen
- Zurückführen an Plätze j und j+n

Parallele Algorithmen

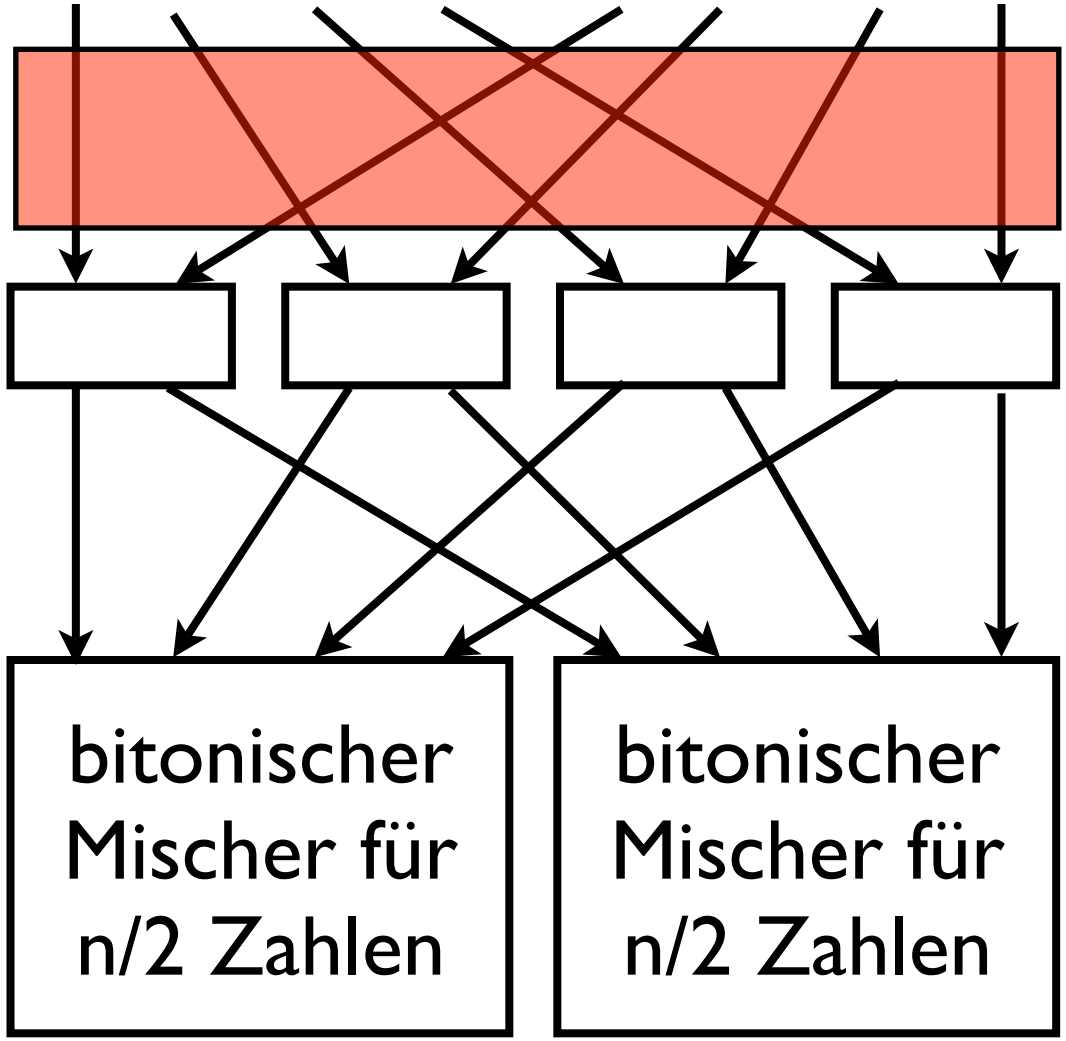
Prozessornetzwerk - Architektur



Parallele Algorithmen

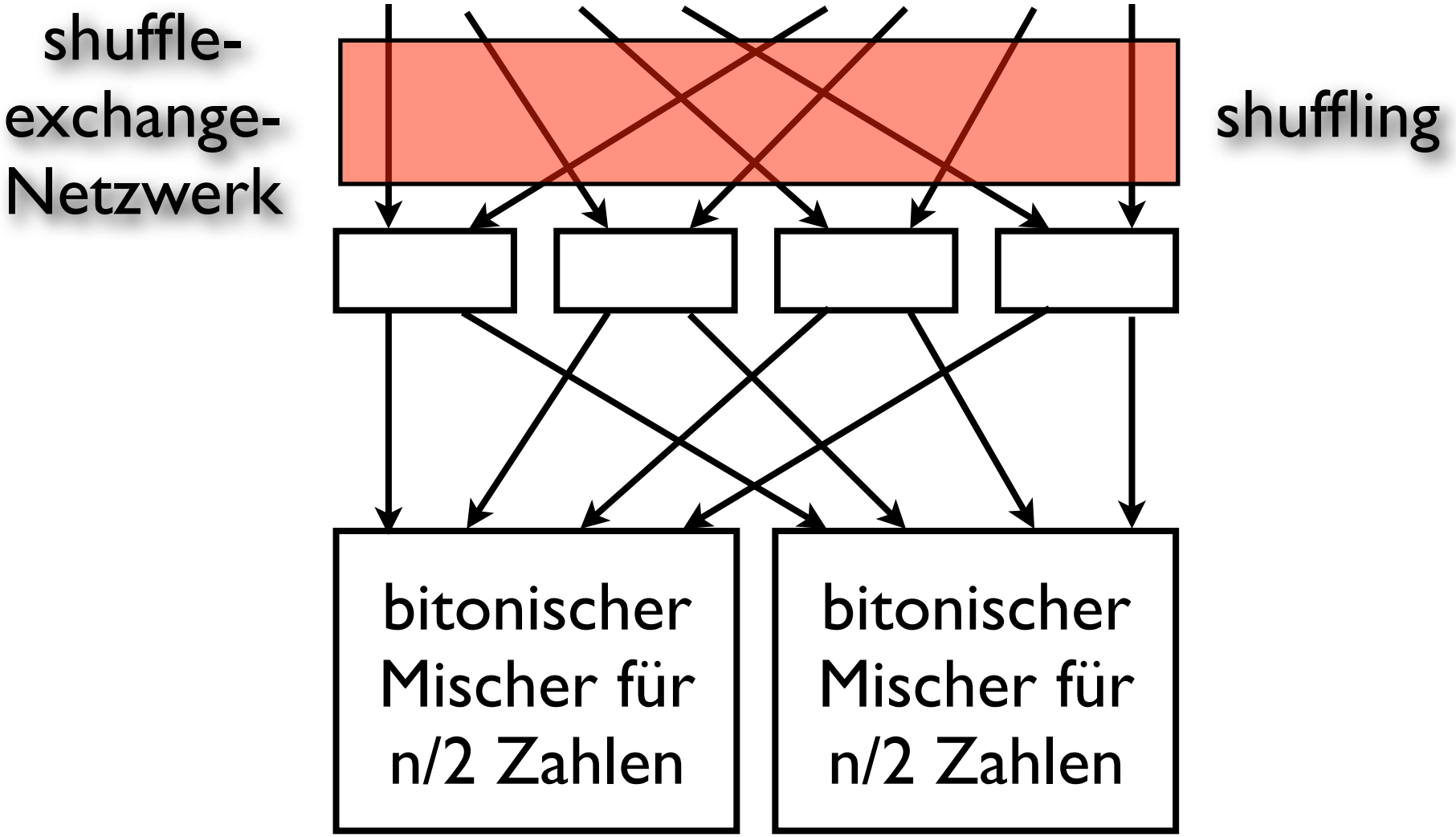
Prozessornetzwerk - Architektur

shuffle-exchange-Netzwerk



Parallele Algorithmen

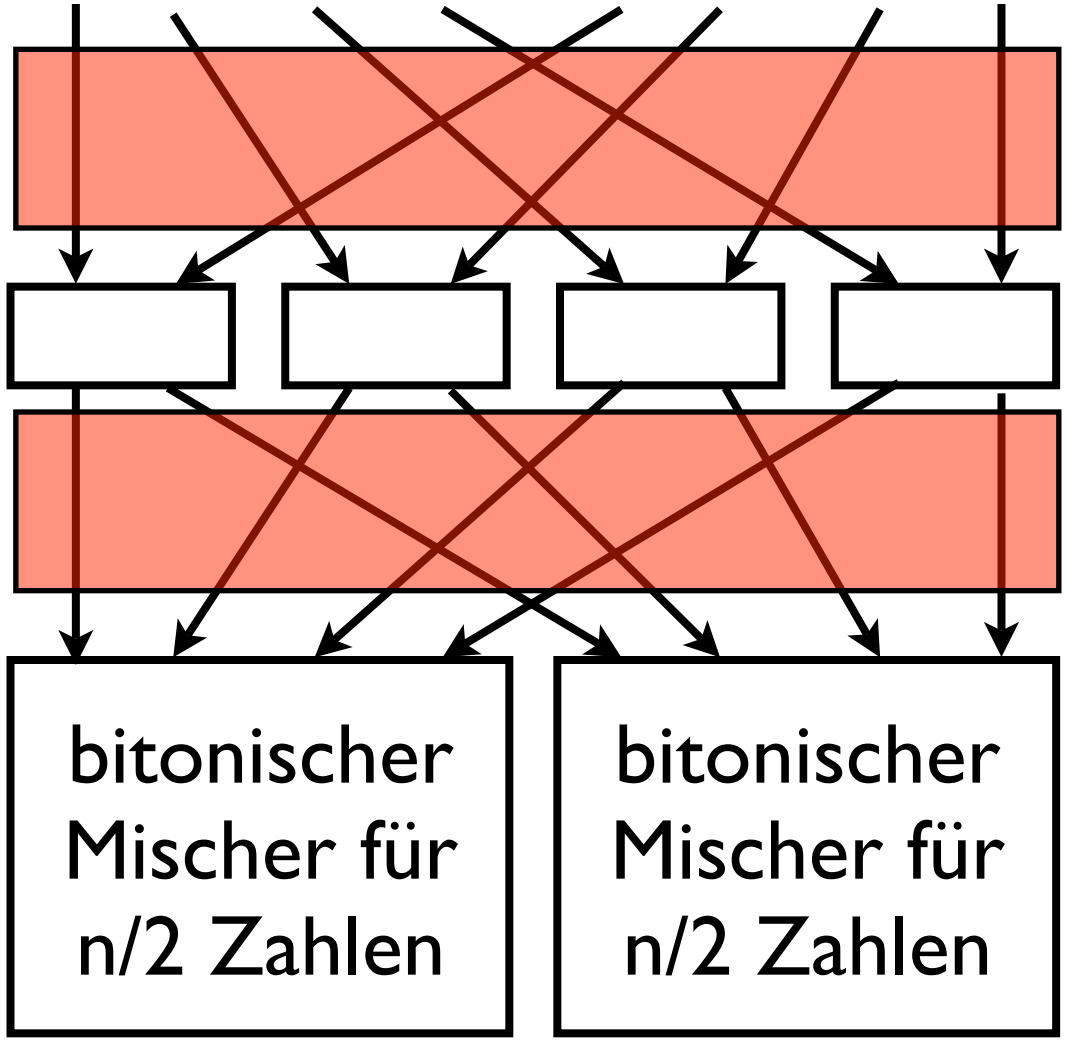
Prozessornetzwerk - Architektur



Parallele Algorithmen

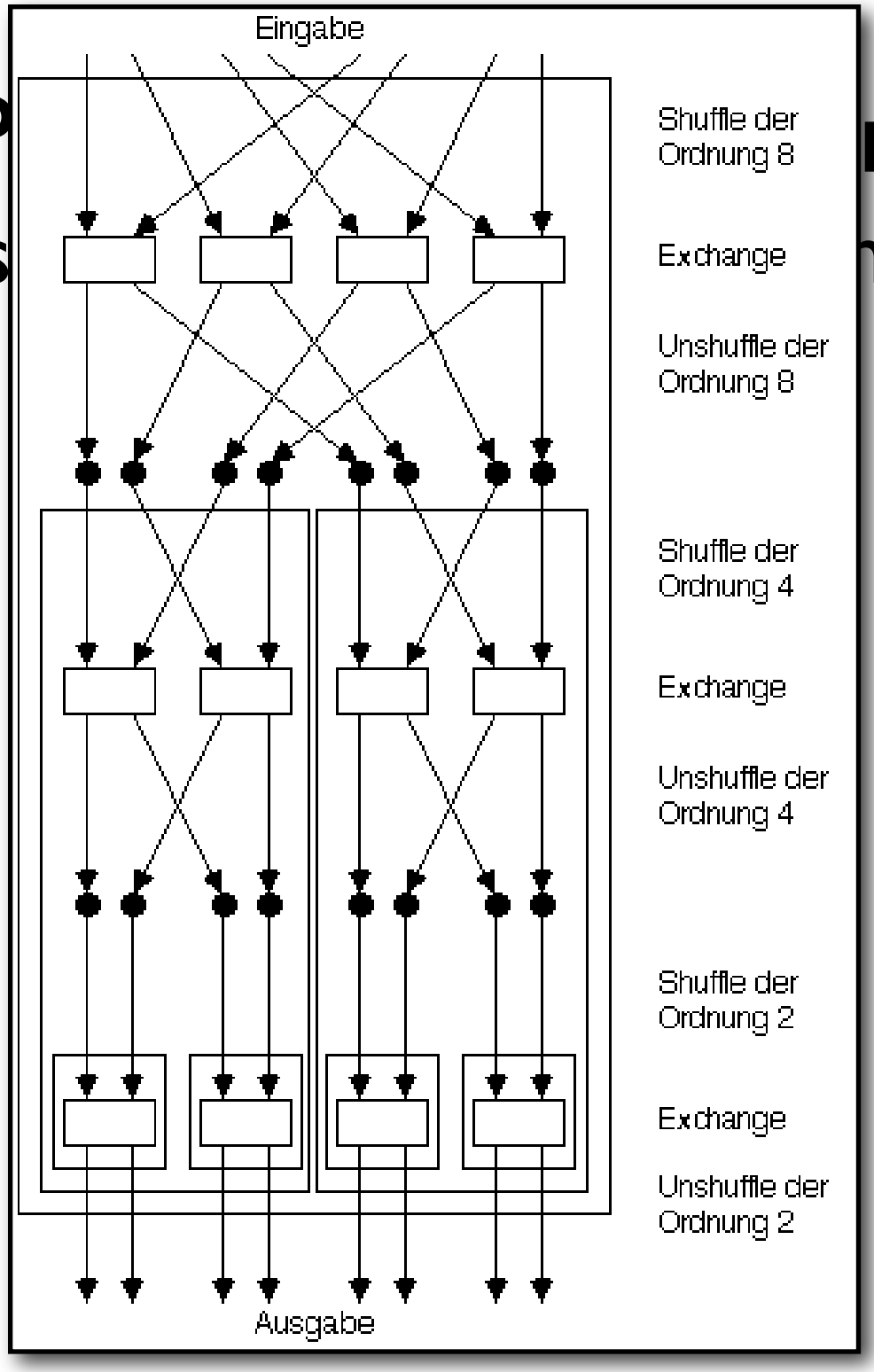
Prozessornetzwerk - Architektur

shuffle-exchange-Netzwerk



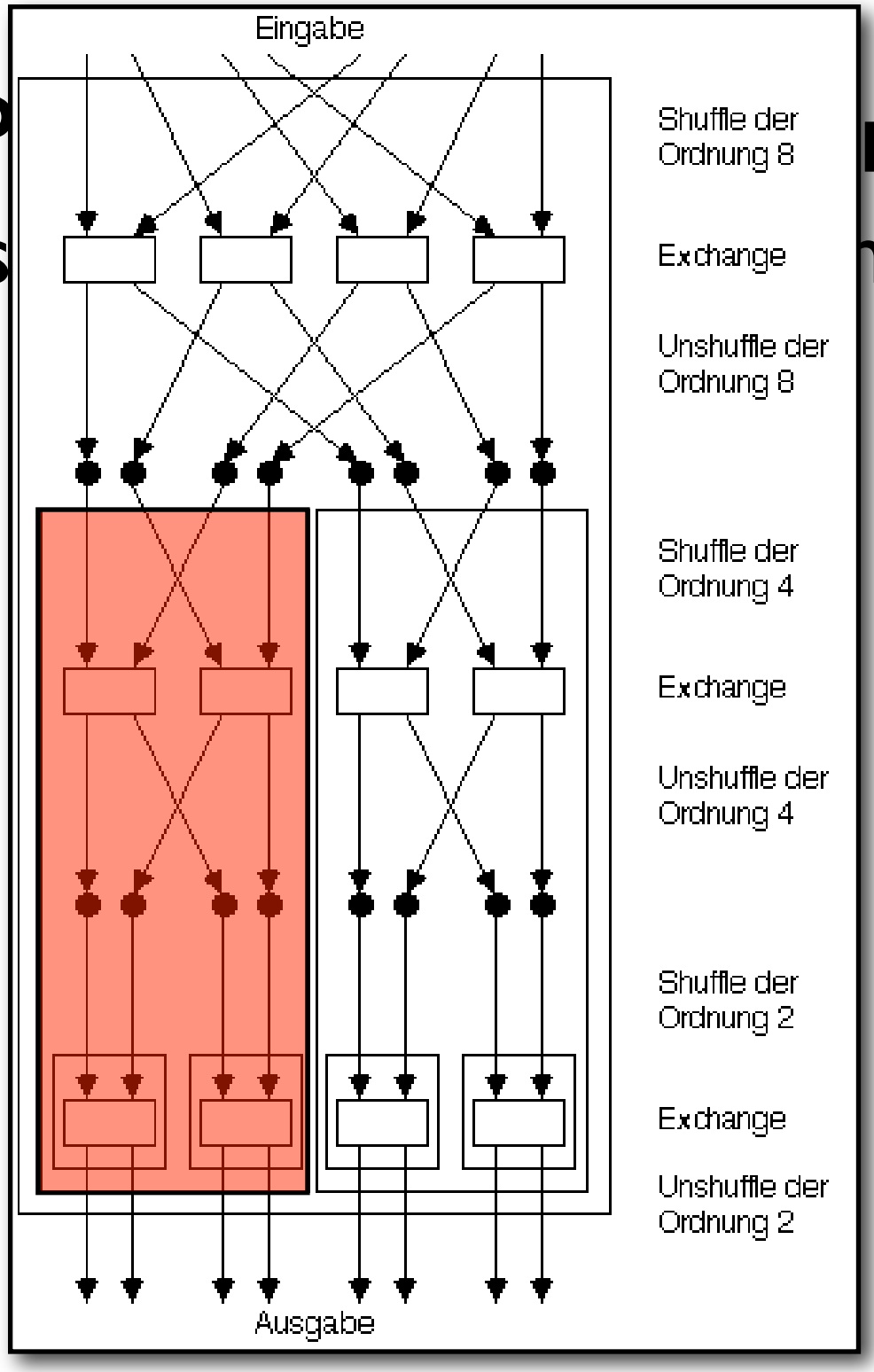
P n

Prozes henbau



P n
Prozes henbau

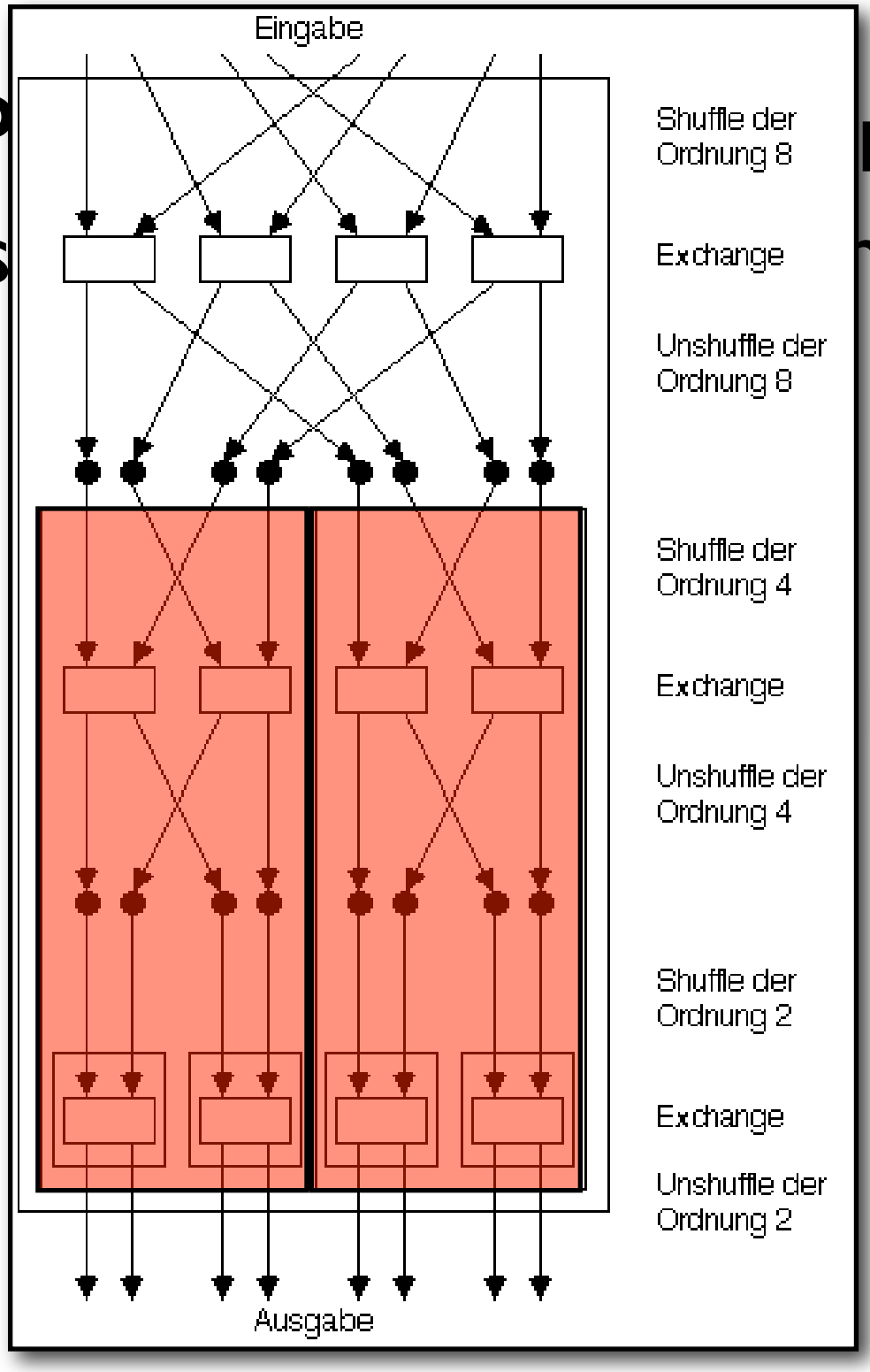
biton. Mischer
für n/2=4
Zahlen



P n

Prozes henbau

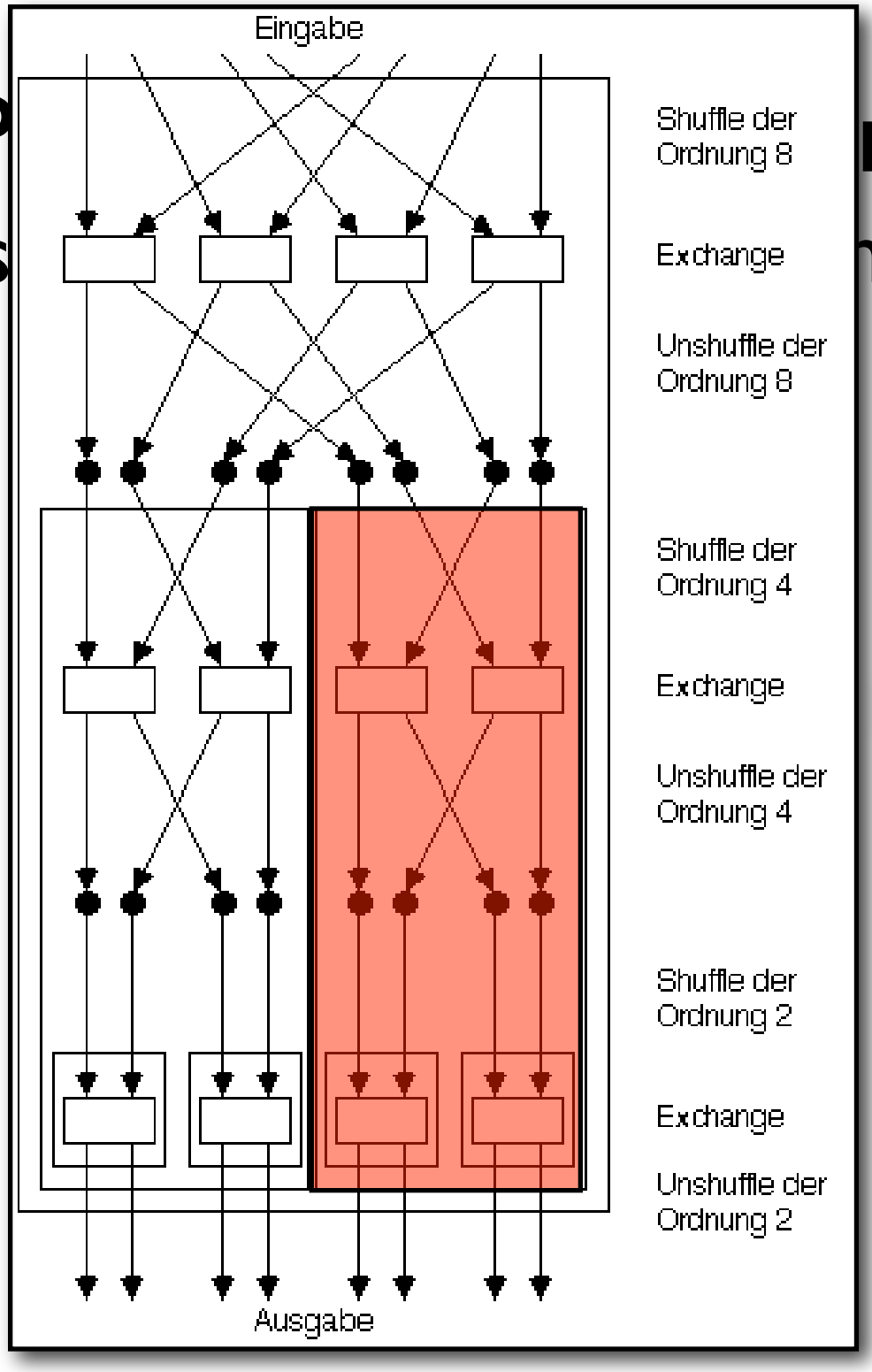
biton. Mischer für n/2=4 Zahlen



biton. Mischer für n/2=4 Zahlen

P n

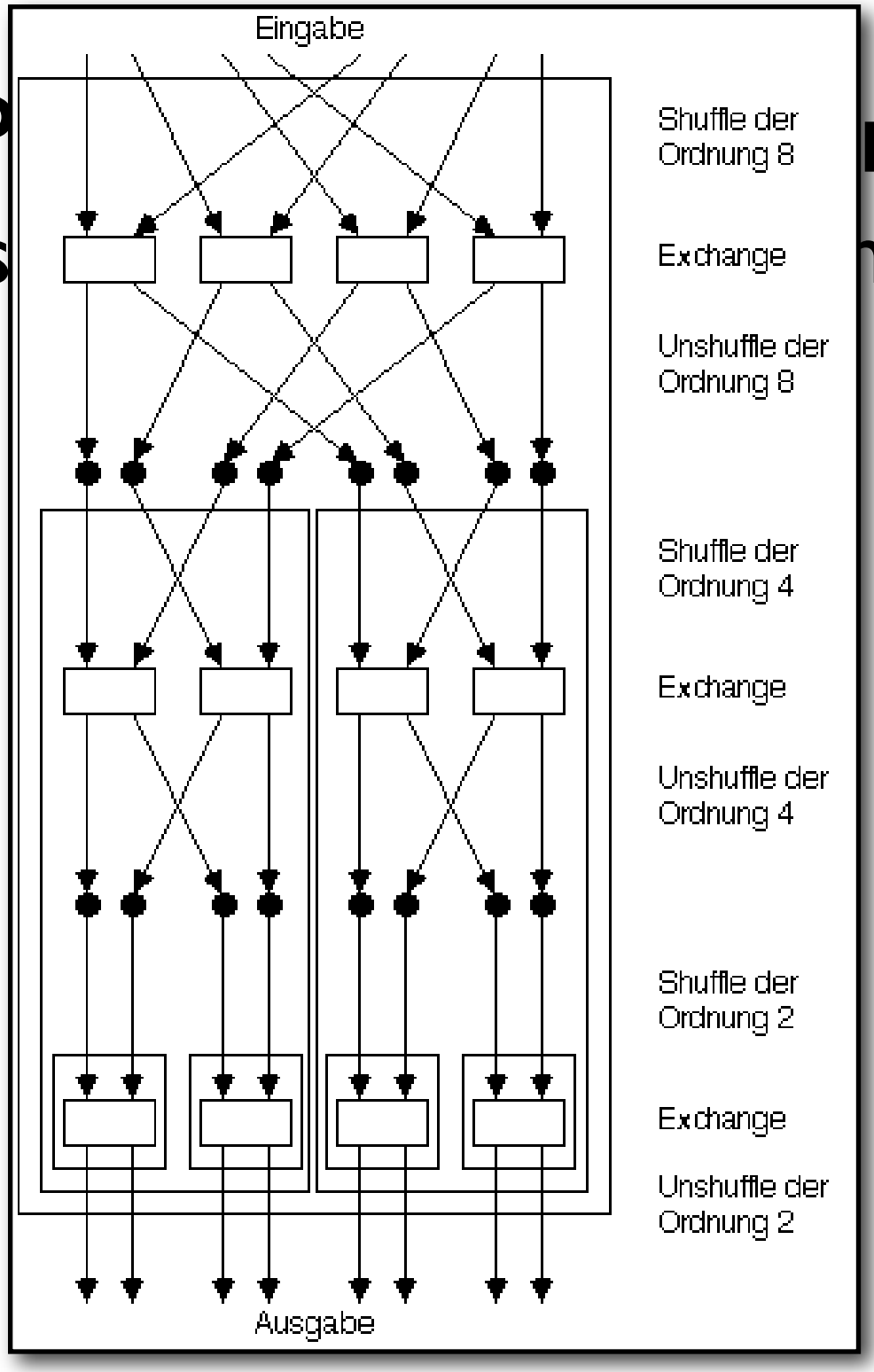
Prozes henbau



biton. Mischer für n/2=4 Zahlen

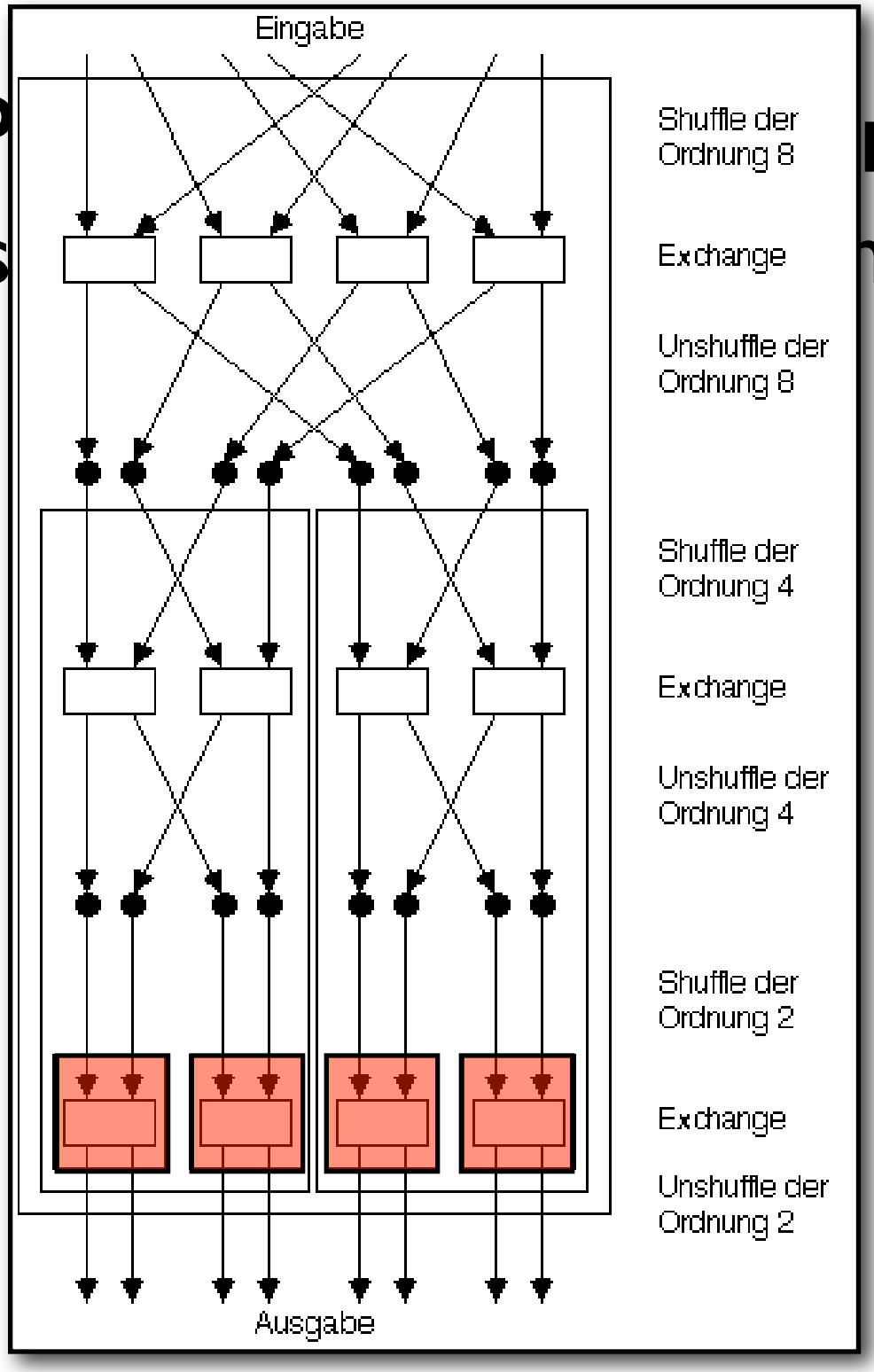
P n

Prozes henbau



P n

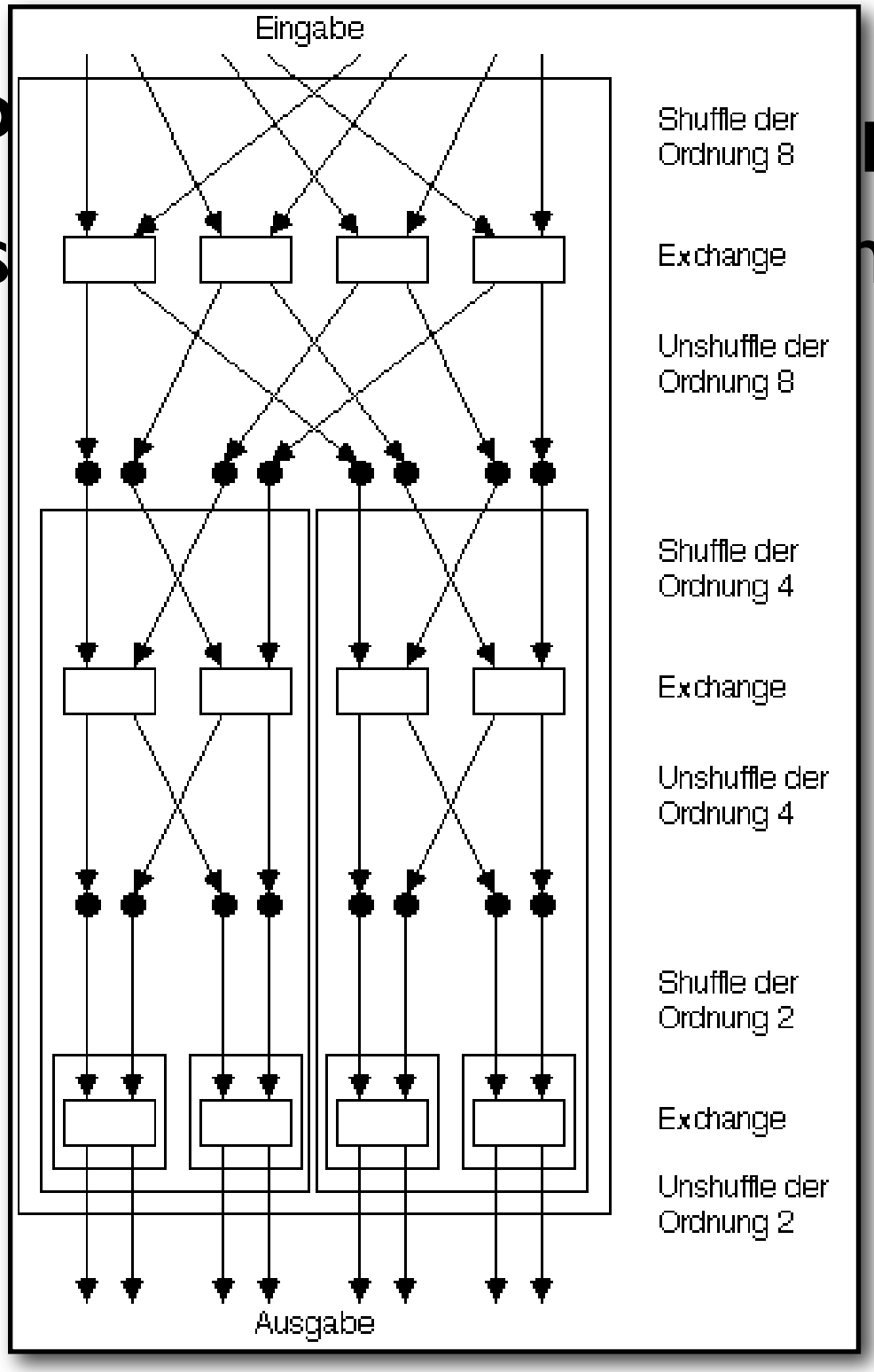
Prozes henbau



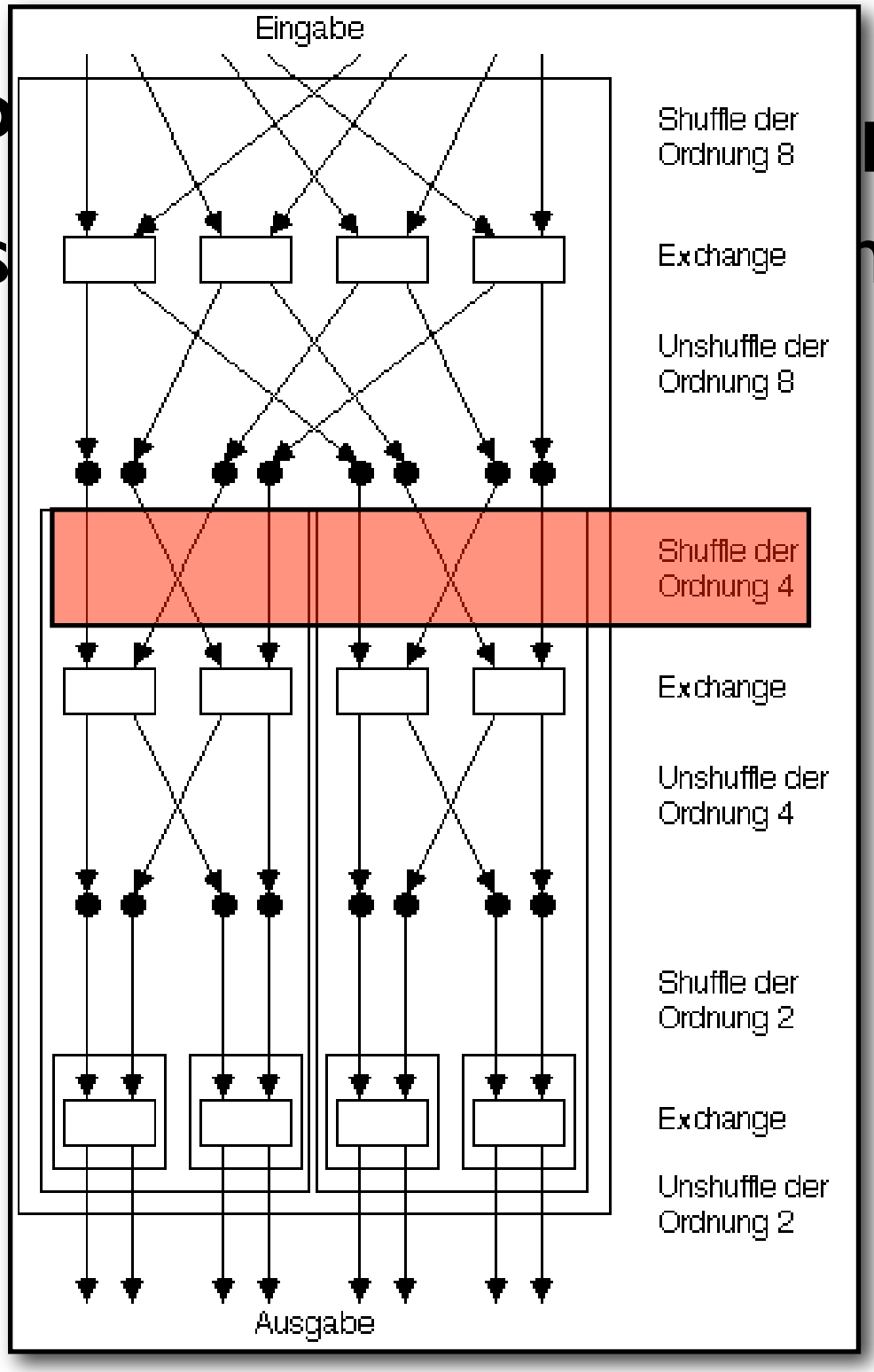
biton. Mischer für n/4=2 Zahlen

P n

Prozes henbau

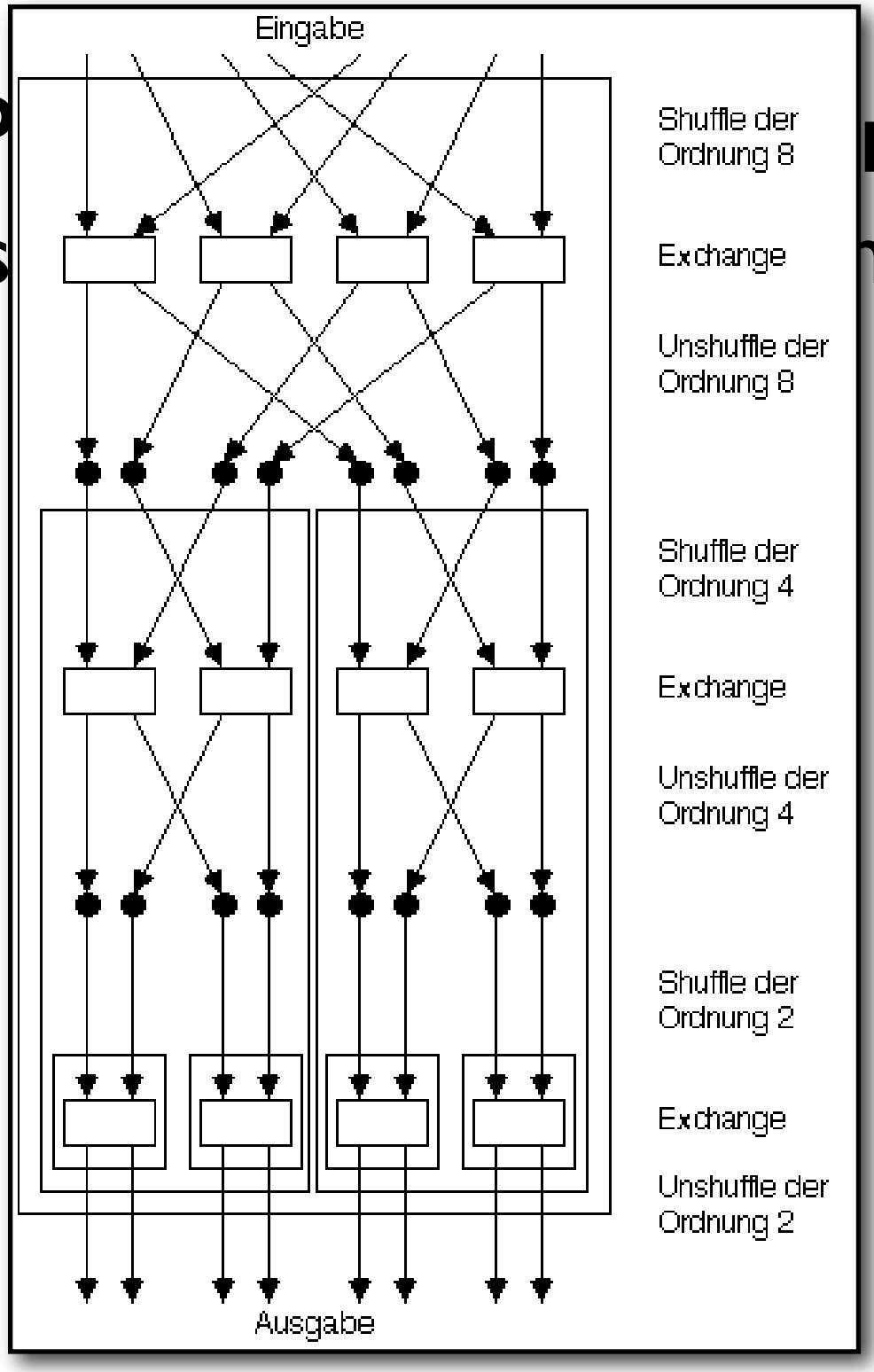


P n
Prozes henbau



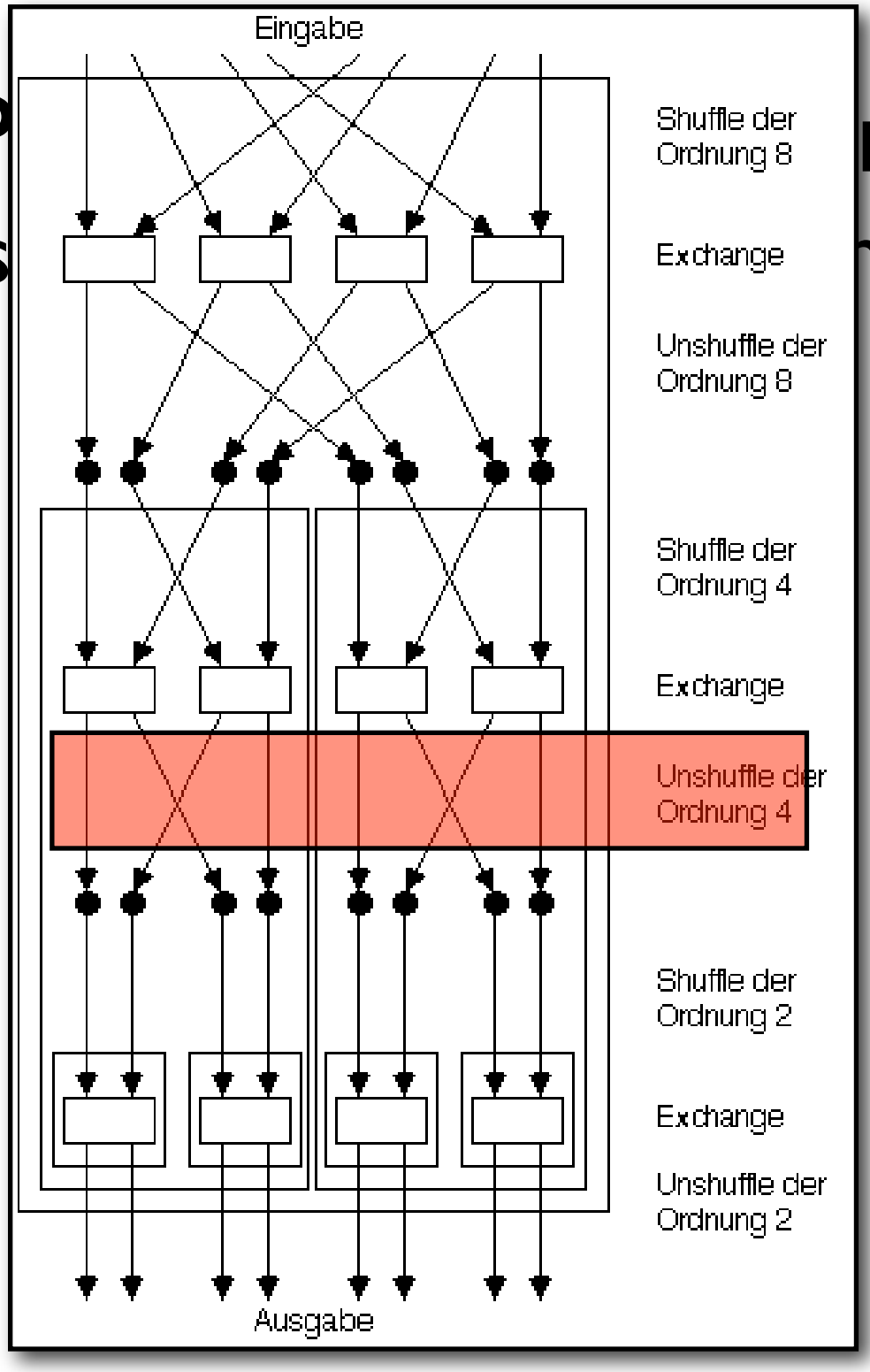
P n

Prozes henbau



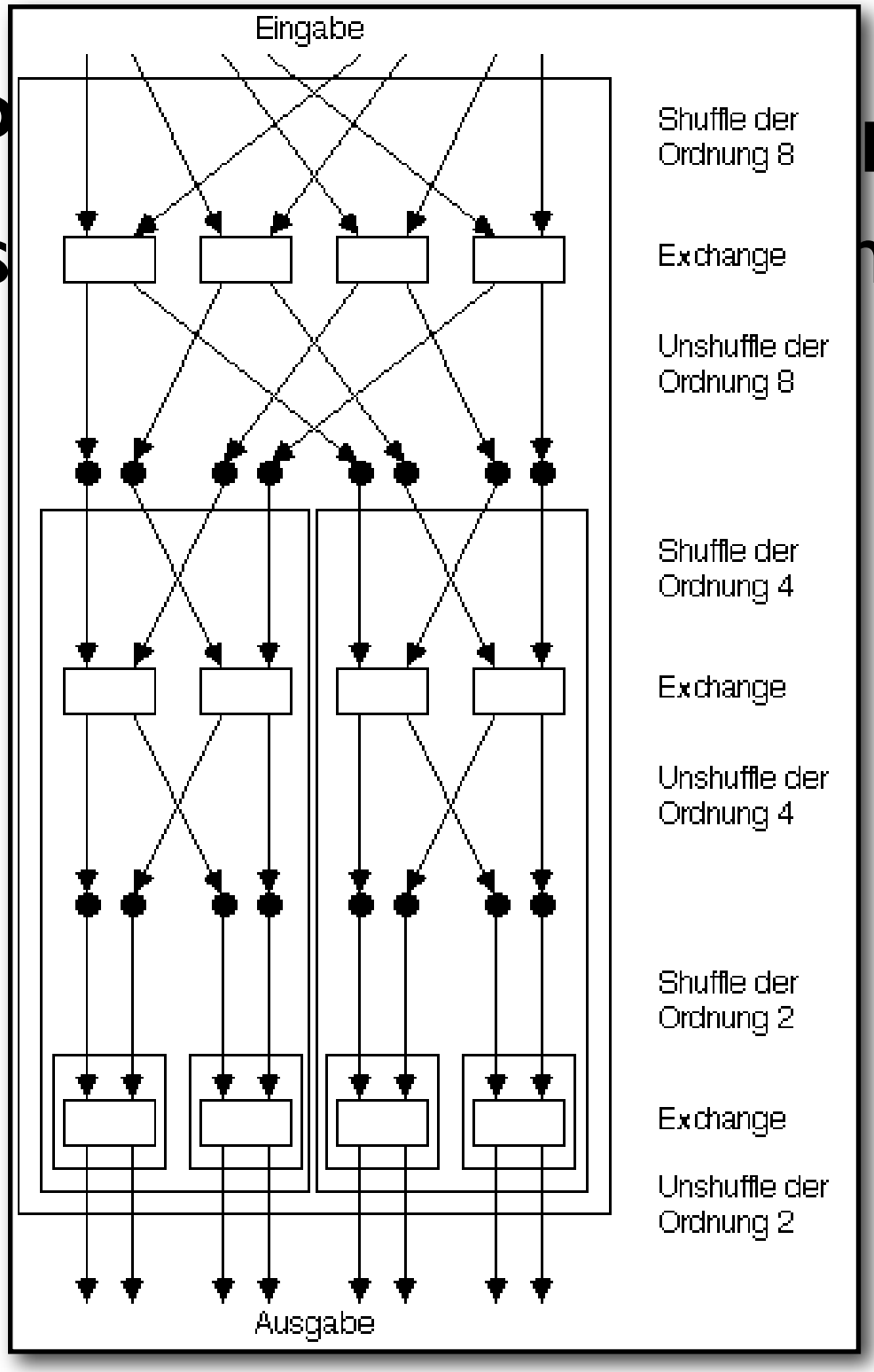
P n

Prozes henbau



P n

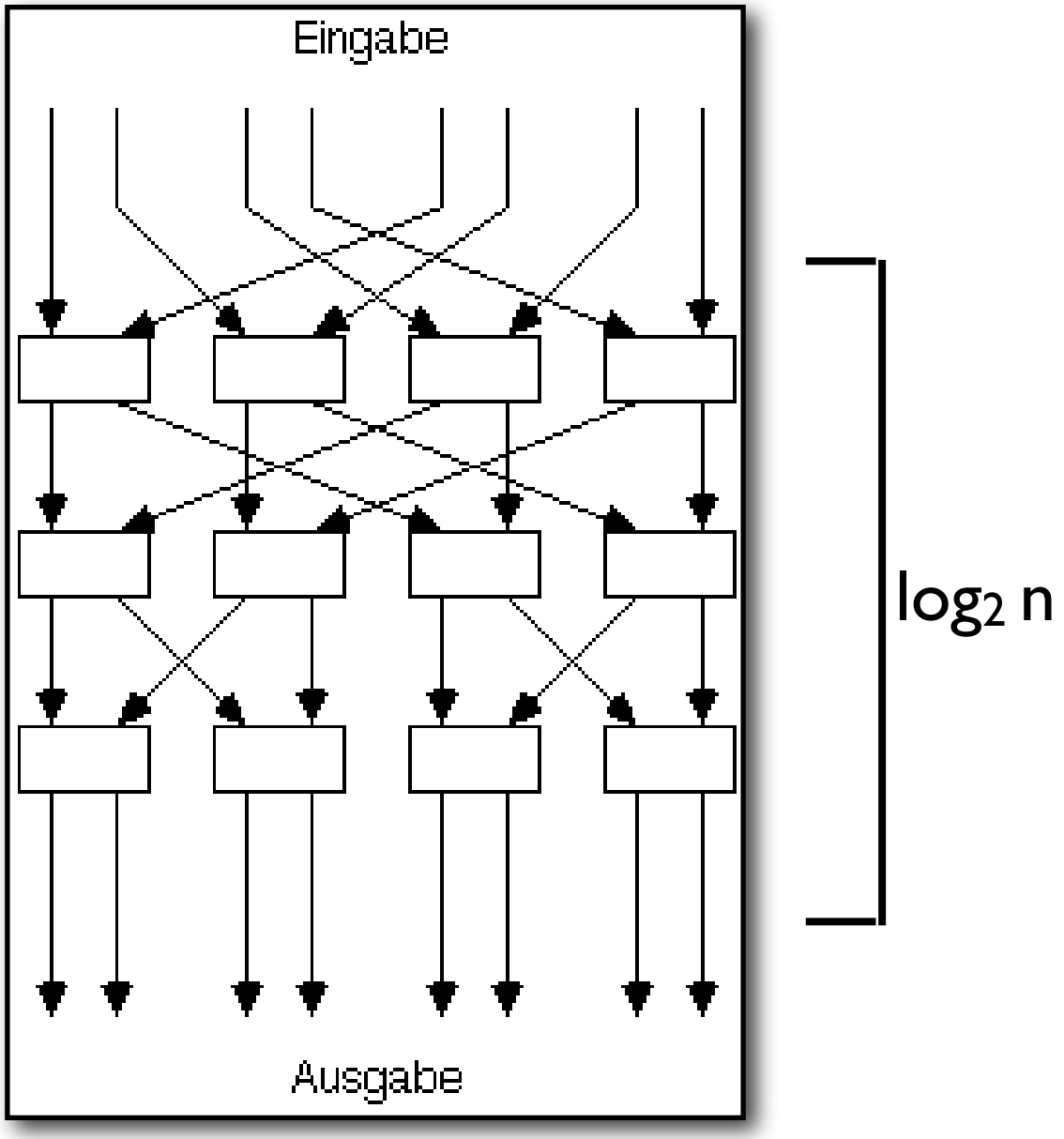
Prozes henbau



Parallele Algorithmen

Prozessornetzwerk - Reduktion

bitonisches Mischen (n=8)



Effizienz

- $\log_2 n$ Shuffle-Exchange-Stufen
- $\log_2 n$ Takte

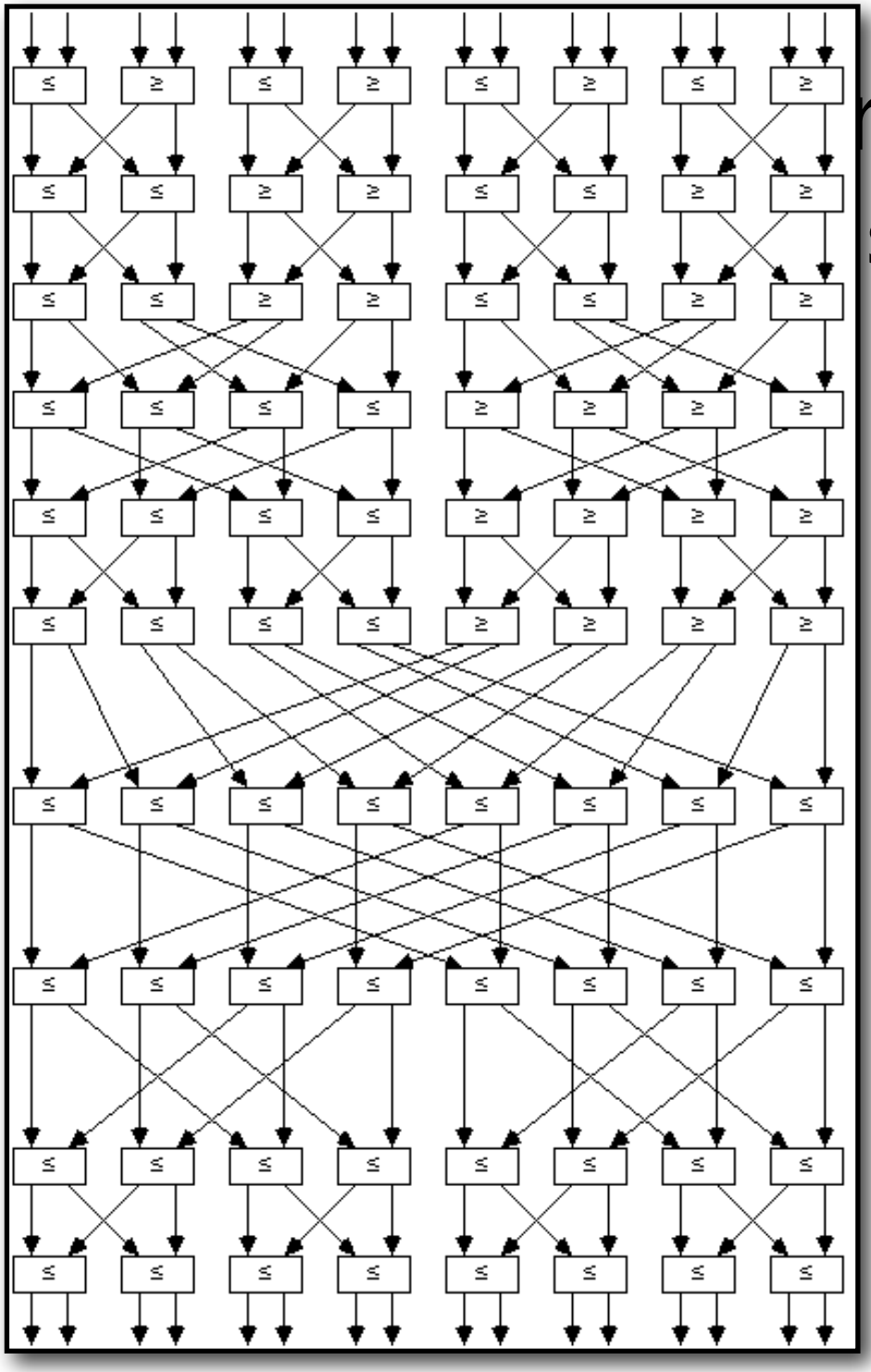
Parallele Algorithmen
Sortiernetzwerk

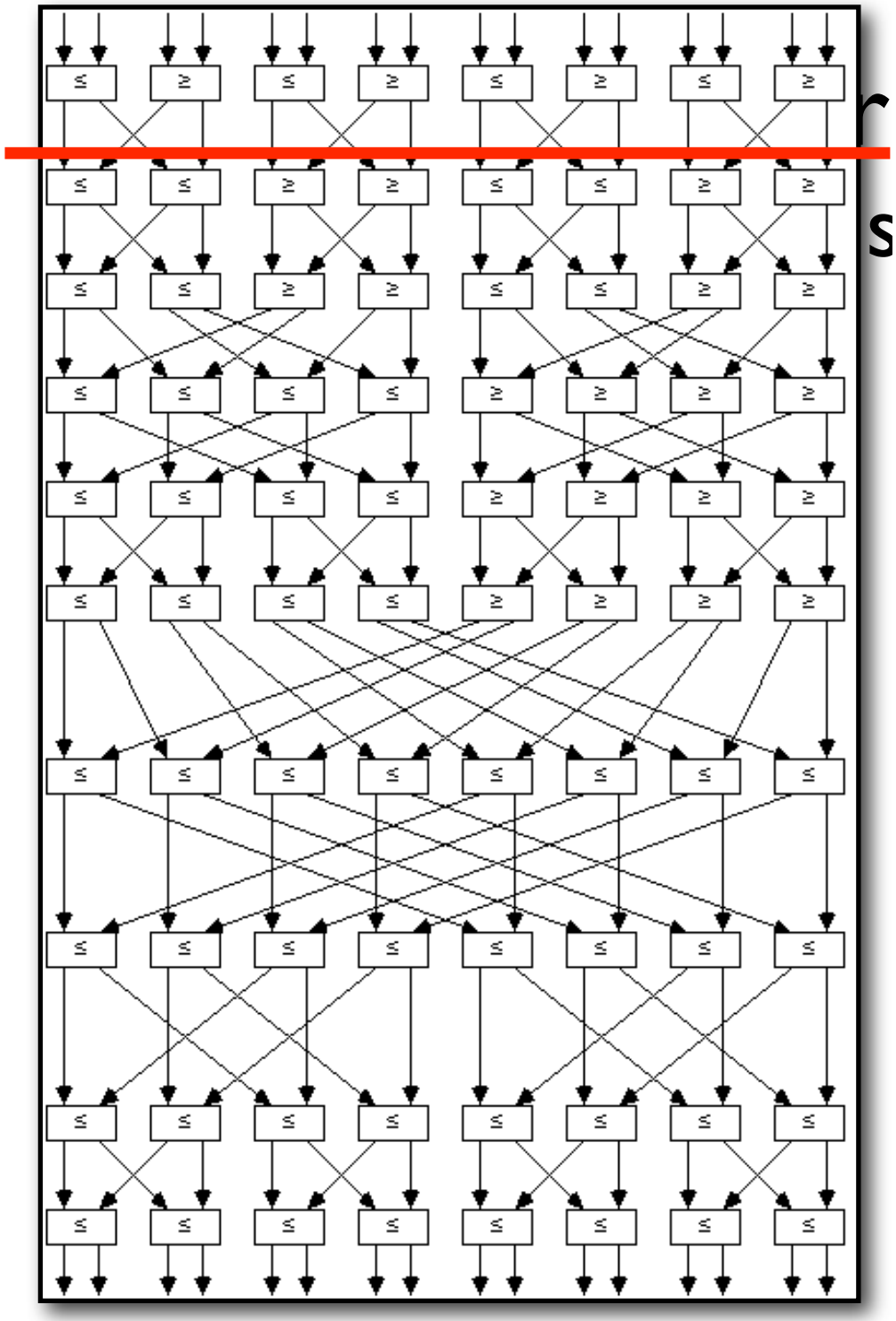
Zusammenbau des Sortiernetzwerks

- Idee: Rekursion
  - 1. Schritt: Mache aus je vier aufeinanderfolgenden Elementen eine bitonische Folge.
    - Ergebnis: n/4 bitonische Folgen der Länge 4.
  - 2. Schritt: Verwende Algorithmus zum bitonischen Mischen und mache aus den n/4 Folgen gleichviele Folgen, die im Wechsel aufsteigend und absteigend sortiert.
    - Ergebnis: n/8 bitonische Folgen der Länge 8.
  - Rekursives Anwenden, bis sortierte Folge vorliegt.

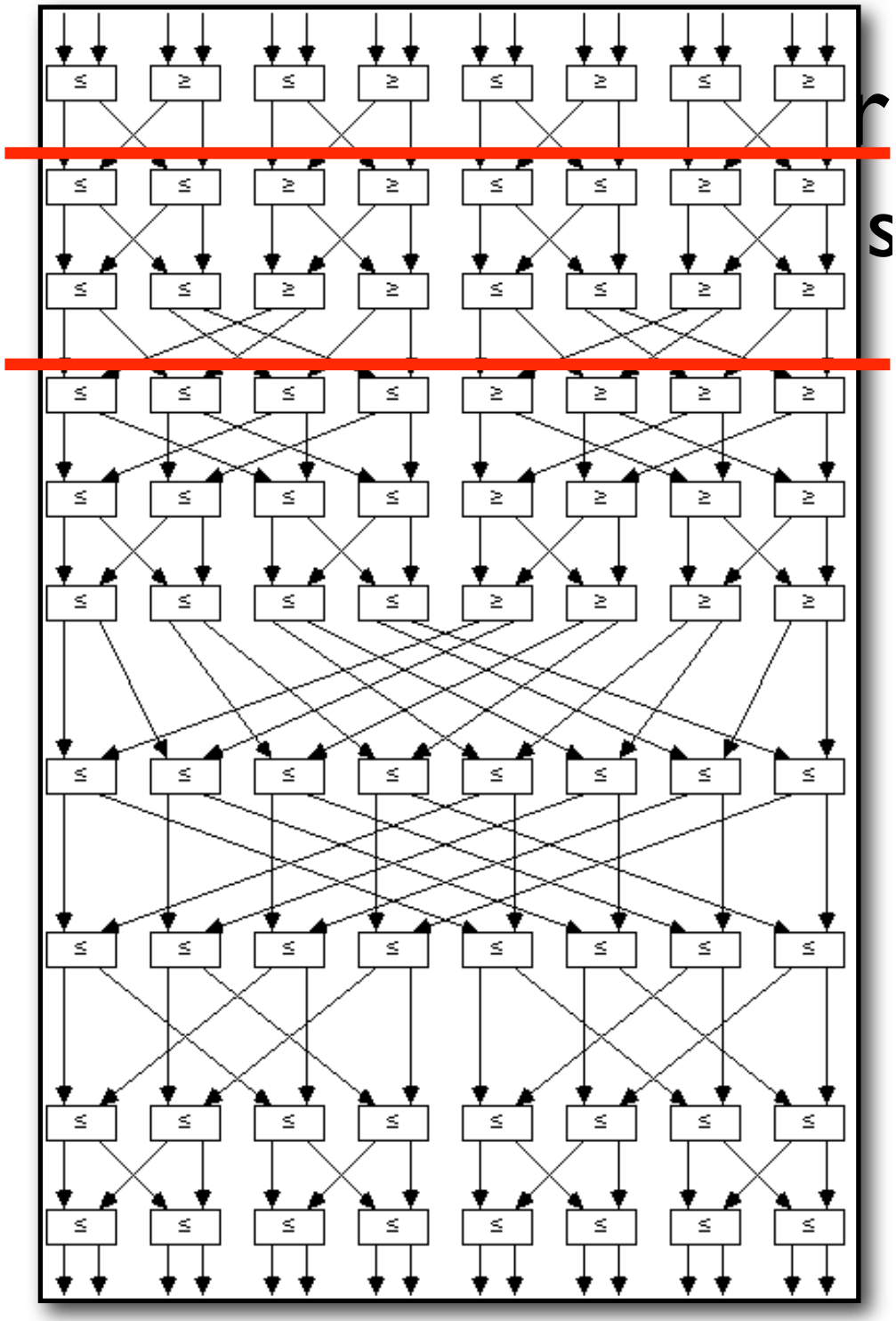
Parallele Algorithmen

Sortiernetzwerk - vollständig (n=16)



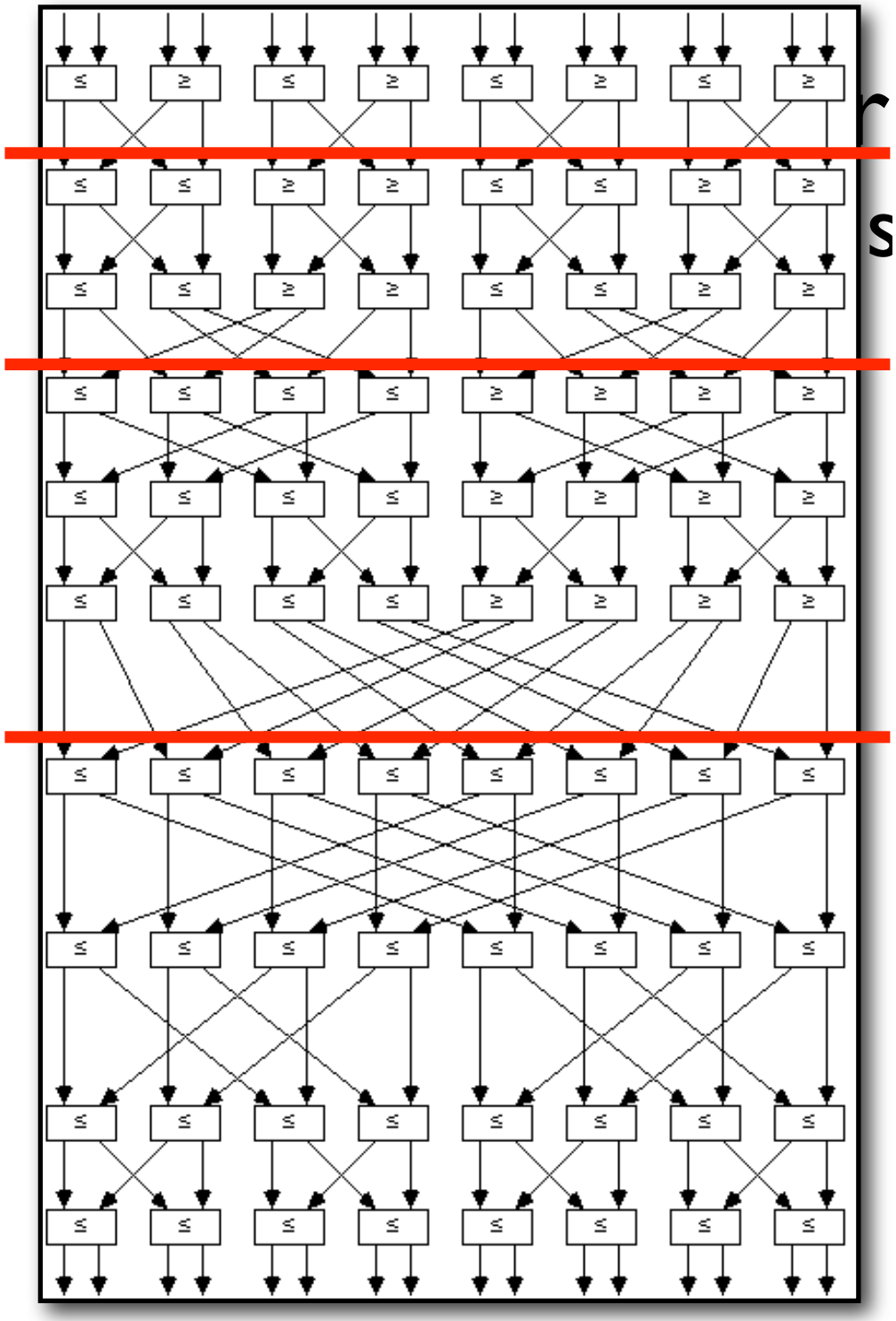


hier liegen 4 bitonische Folgen der Länge 4 vor



hier liegen 4 bitonische Folgen der Länge 4 vor

hier liegen 2 bitonische Folgen der Länge 8 vor



hier liegen 4 bitonische Folgen der Länge 4 vor

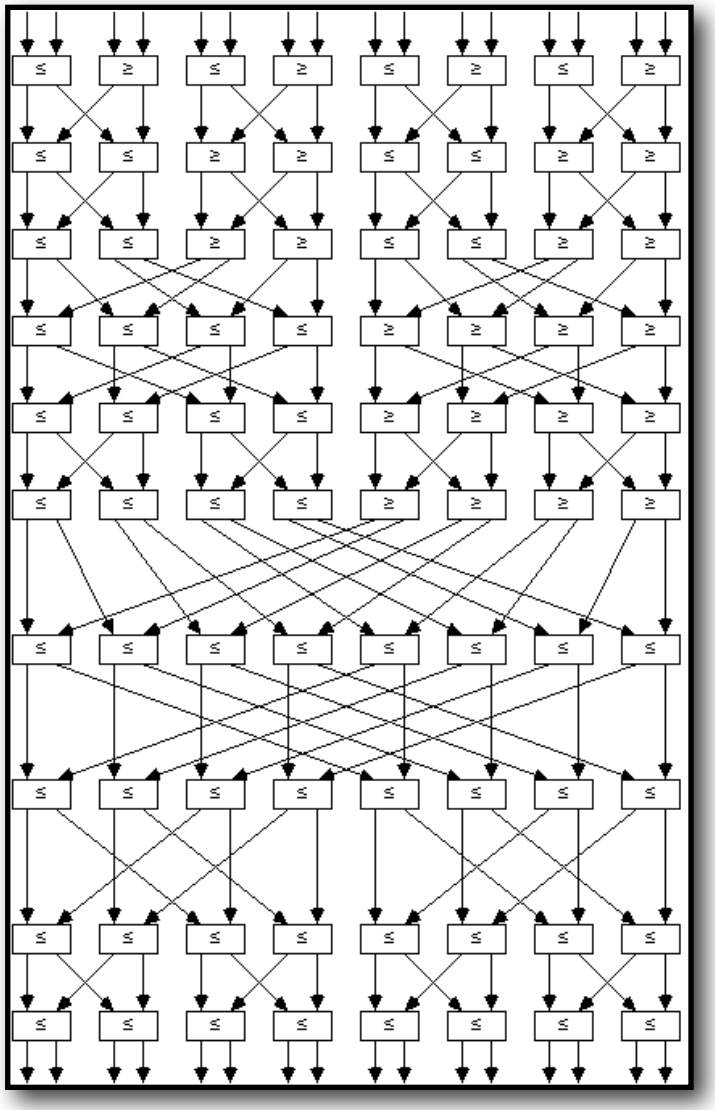
hier liegen 2 bitonische Folgen der Länge 8 vor

hier liegt 1 bitonische Folgen der Länge 16 vor

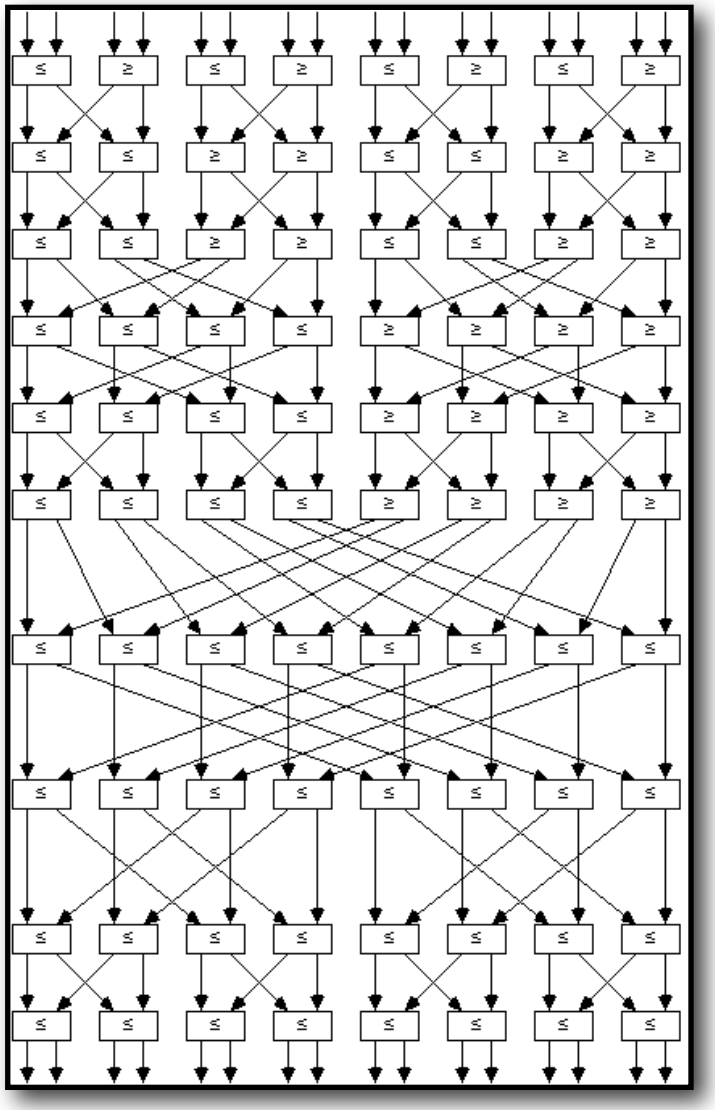
Parallele Algorithmen

Sortiernetzwerk - Effizienz

Parallele Algorithmen
Sortiernetzwerk - Effizienz

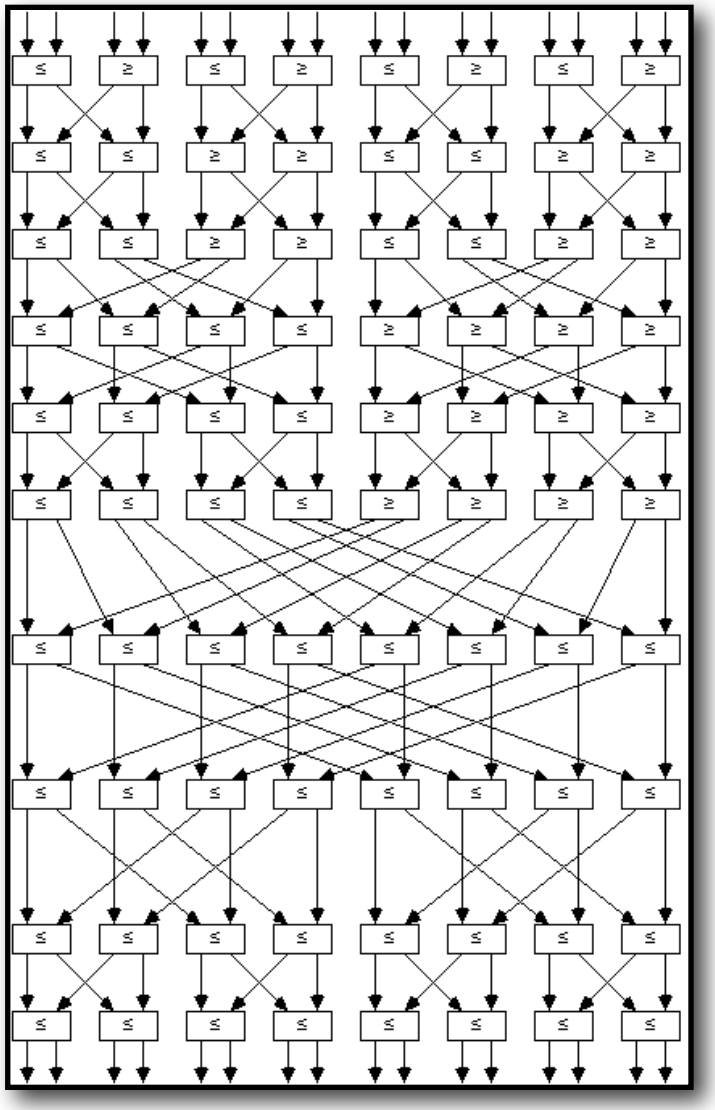


Parallele Algorithmen
Sortiernetzwerk - Effizienz



Mischen: $\log_2 n$ Takte

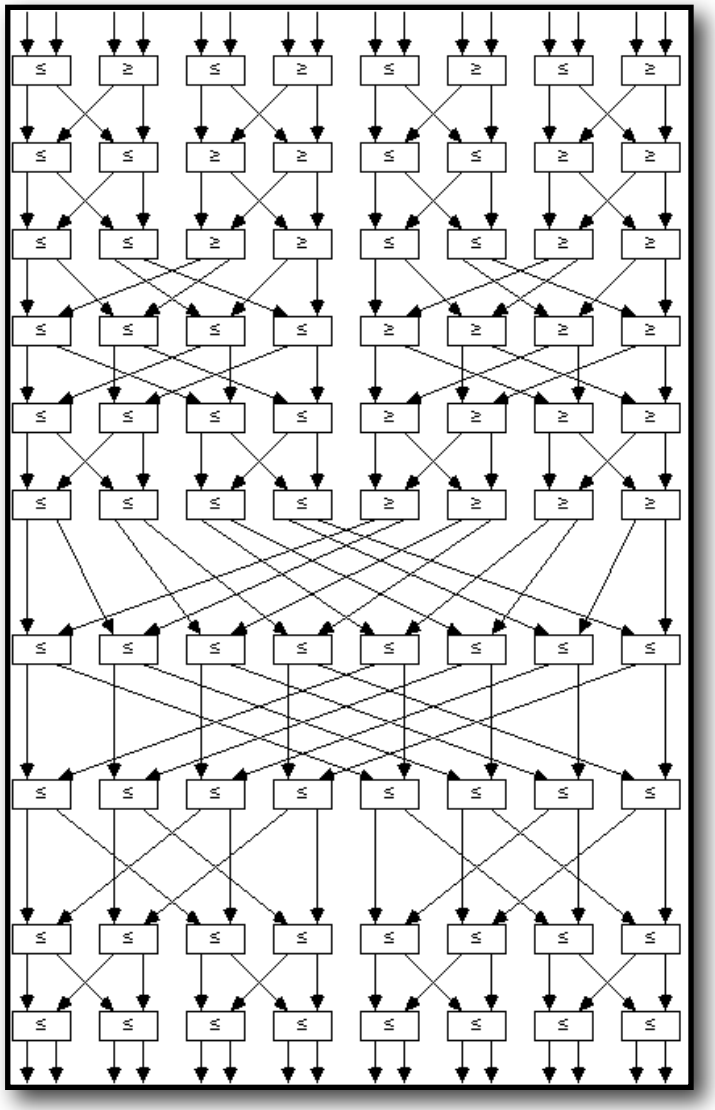
Parallele Algorithmen
Sortiernetzwerk - Effizienz



n/4 bitonische Folgen der Länge 4: 1 Takt

Mischen: $\log_2 n$ Takte

Parallele Algorithmen
Sortiernetzwerk - Effizienz

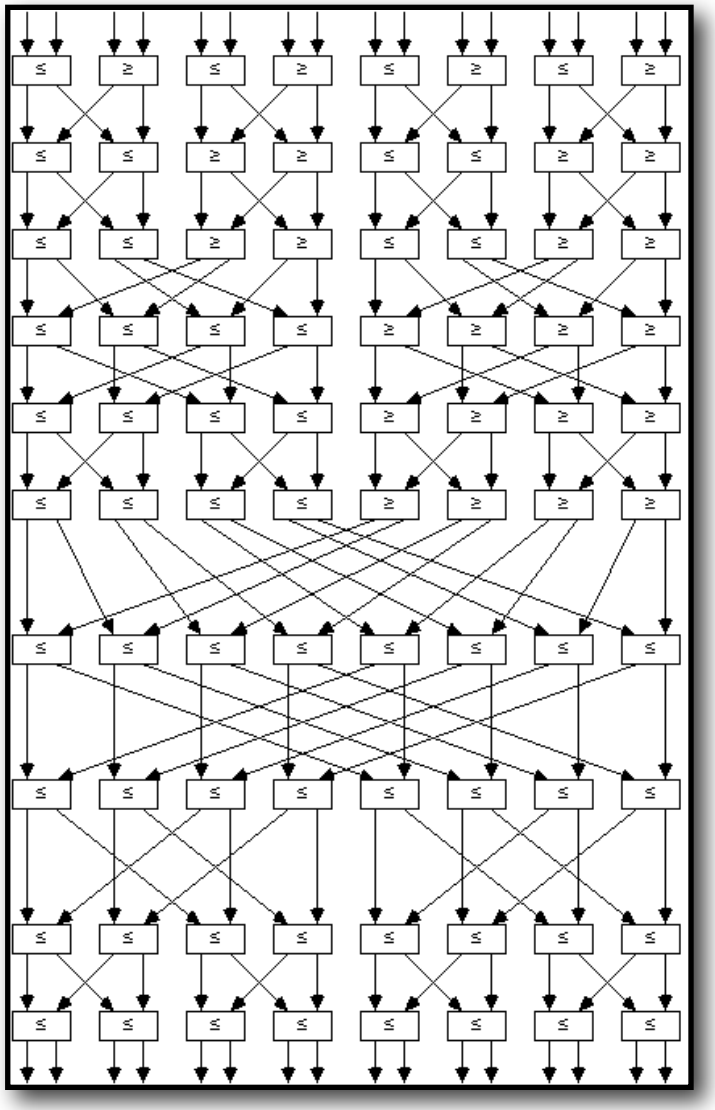


n/4 bitonische Folgen der Länge 4: 1 Takt

n/8 bitonische Folgen der Länge 8: 2 Takte

Mischen: $\log_2 n$ Takte

Parallele Algorithmen
Sortiernetzwerk - Effizienz



n/4 bitonische Folgen der Länge 4: 1 Takt

n/8 bitonische Folgen der Länge 8: 2 Takte

n/16 bitonische Folgen der Länge 16: 3 Takte

Mischen: $\log_2 n$ Takte

Parallele Algorithmen
Sortiernetzwerk - Effizienz

Algorithmusteil	**Takte**
letzter Schritt: Mischen einer bitonischen Folge	$O(\log_2 n)$
Herstellen einer bitonischen Folge	
1. Schritt: Erzeuge n/4 bitonische Folgen der Länge 4	1
2. Schritt: Erzeuge n/8 bitonische Folgen der Länge 8	2
3. Schritt: Erzeuge n/16 bitonische Folgen der Länge 16	3
...	
k. Schritt: Erzeuge $n/2^{k+1}$ bitonische Folgen der Länge 2^{k+1}	k
...	
Summe	T(n)

Parallele Algorithmen
Sortiernetzwerk - Effizienz

$$
\begin{aligned}
T(n) &= 1+2+3+\ldots+(\log_2 n-1)+\log_2 n \\
&= 1/2(\log_2 n)(\log_2 n+1) \\
&\approx 1/2(\log_2 n)^2 = O((\log_2 n)^2).
\end{aligned}
$$

Anzahl Prozessoren: $O(n(\log_2 n)^2)$.

Aber: n Prozessoren reichen, wenn man Output-Leitungen auf Input schaltet.

Problem: wie realisiert man dynamisch die Shuffle-Vernetzung?

Parallele Algorithmen
Sortiernetzwerk - Effizienzvergleich

- Effizienz von parallelen Algorithmen bewertet man in Relation zu den sequentiellen:
  - Sequentieller Fall mit einem Prozessor: Beste Laufzeit f(n)
- Paralleler Fall mit p Prozessoren: Laufzeit g(n)
  - Erwartete Laufzeit f(n)/p
  - **Speed-up** f(n)/g(n)
  - paralleler Algorithmus **optimal**, wenn f(n)/p, d.h. Speed-up von p erreicht wird

Parallele Algorithmen
Sortiernetzwerk - Speed-up/Effizienz

- $T_p(n)$: Laufzeit eines Algorithmus, der zur Ausführung p Prozessoren benötigt
- **Speed-up** $S_p(n) = T_1(n)/T_p(n)$
  - Offenbar: $1 \leq S_p(n) \leq p$
- **Effizienz** $E_p(n) = S_p(n)/p$ (Quotient aus erreichtem und theoretisch möglichem Speed-up)
  - Offenbar: $1/p \leq E_p(n) \leq 1$.
  - $E_p(n) = 1$, dann Algorithmus **optimal**.

Parallele Algorithmen
Sortiernetzwerk - Effizienzvergleich

- Anwendung auf Sortieren:
  - Sequentieller Fall mit einem Prozessor: Schnellstes Sortierverfahren:

$$O(n \log_2 n)$$

  - Speed-up:

$$O((n \log_2 n)/(\log_2 n)^2)=O(n/\log_2 n)$$

  - Effizienz: $O((n/\log_2 n)/n)=O(1/\log_2 n)$
  - Sortiernetzwerk *nicht* optimal !

Parallele Algorithmen
Sortiernetzwerk - Effizienzvergleich

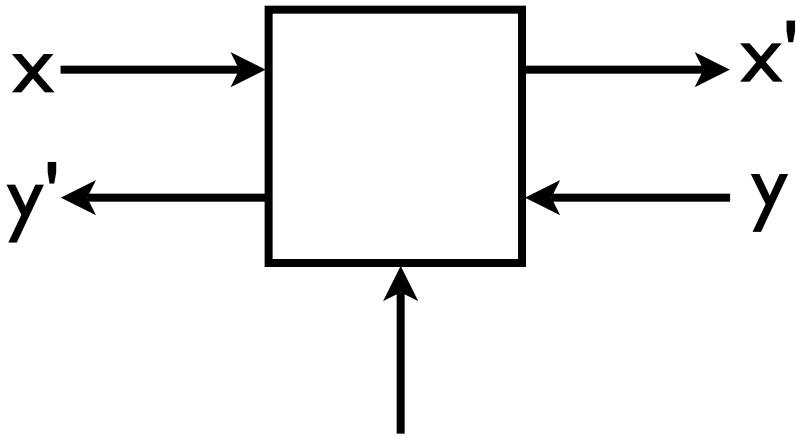
- M. Ajtai, J. Komlós, E. Szemerédi (Ungarn, USA, 1983): paralleler Algorithmus mit n Prozessoren und $O(\log_2 n)$ Takten
- Sehr aufwendiger Beweis !
- sehr große Konstanten (versteckt in Groß-O)

Parallele Algorithmen
Teilworterkennung

- Gesucht: paralleler Algorithmus, der testet, ob ein eingegebenes Wort w ein bestimmtes Teilwort t enthält
- hier: w beliebig, $t=t_1t_2t_3$ (Länge 3)

Parallele Algorithmen

Teilworterkennung - Prozessortyp

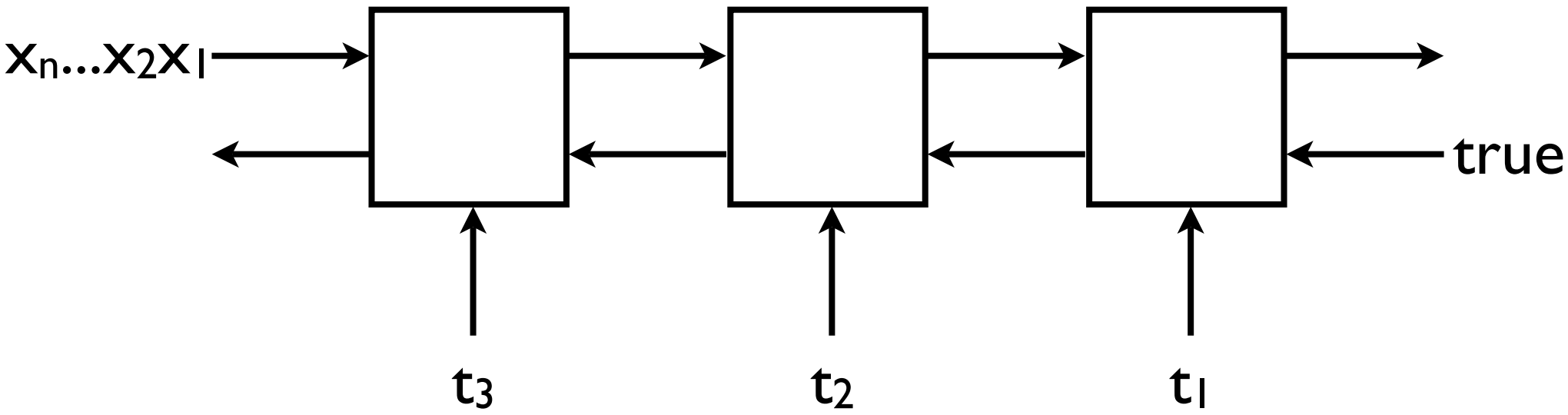


$x'=x$

$y'=y \wedge (x=t_i)$

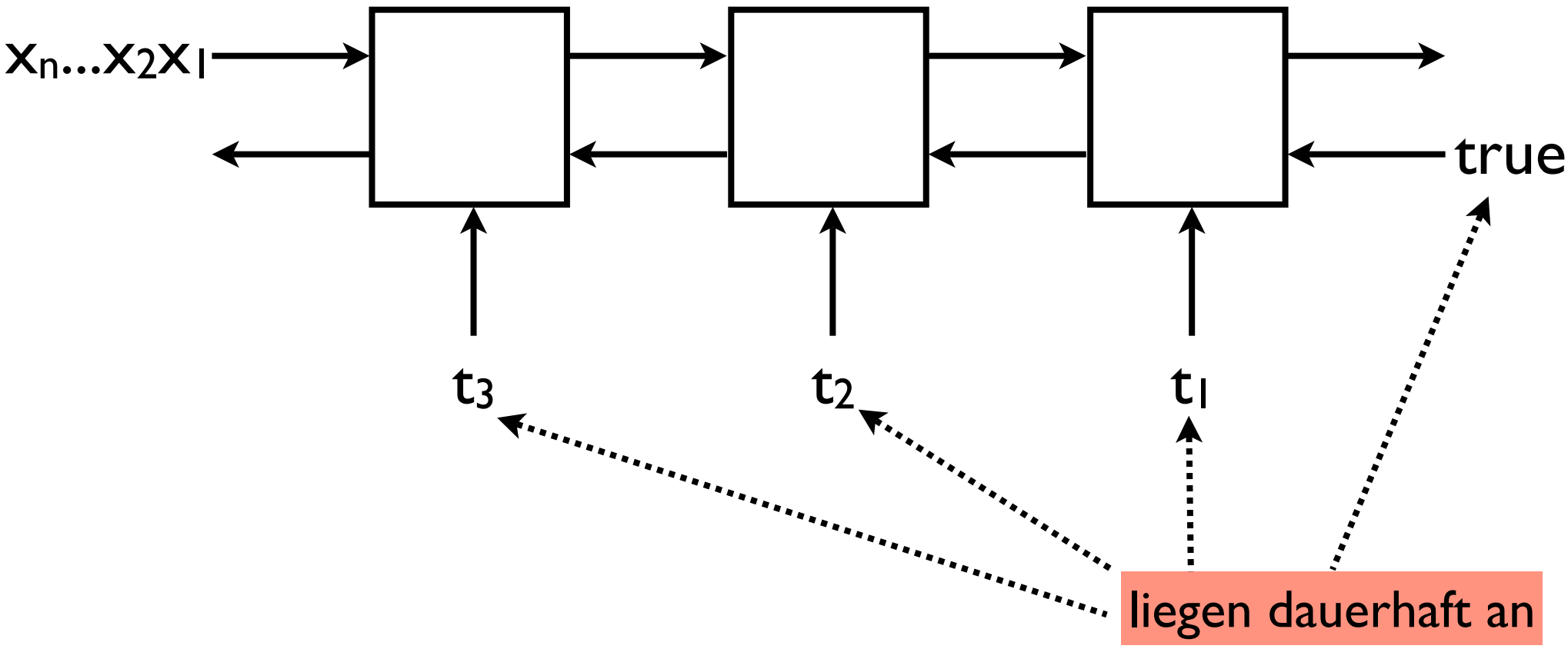
Parallele Algorithmen

Teilworterkennung - Prozessornetz



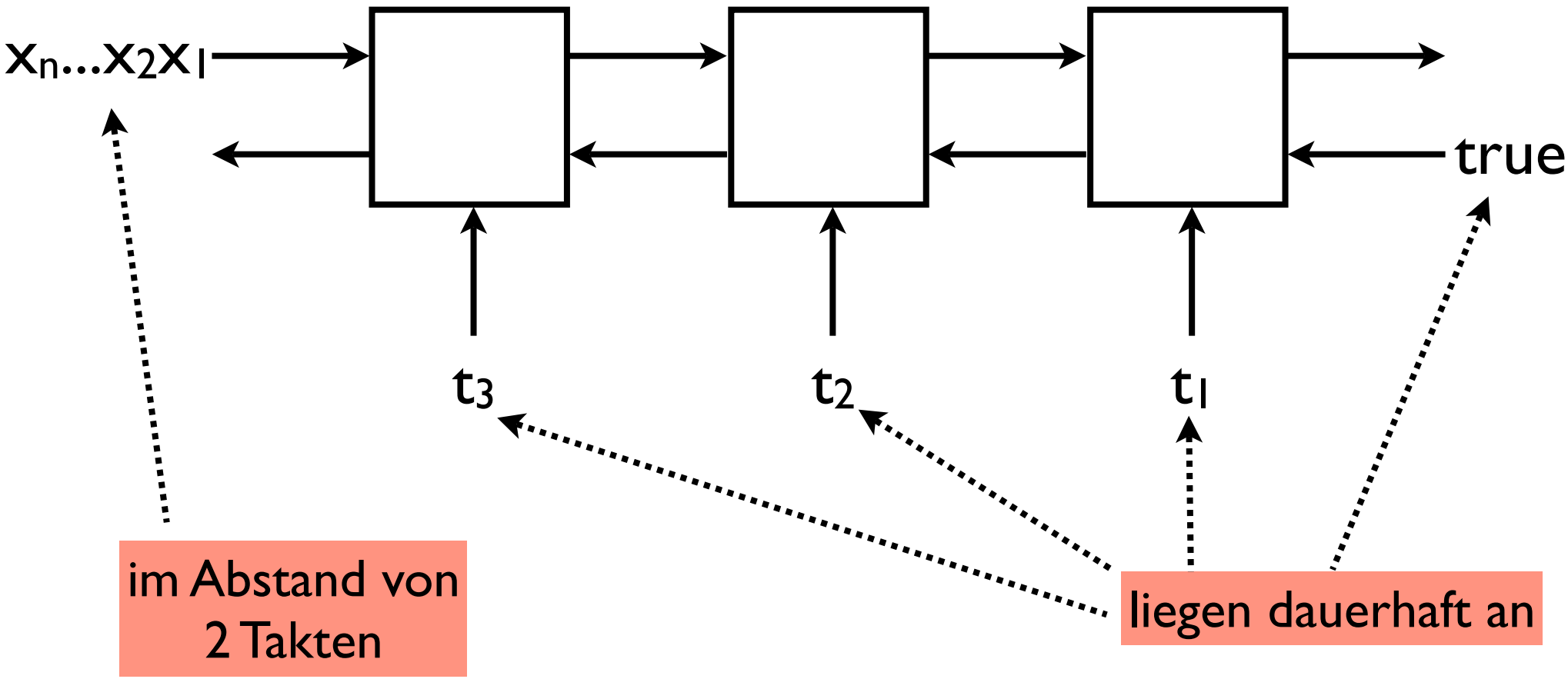
Parallele Algorithmen

Teilworterkennung - Prozessornetz



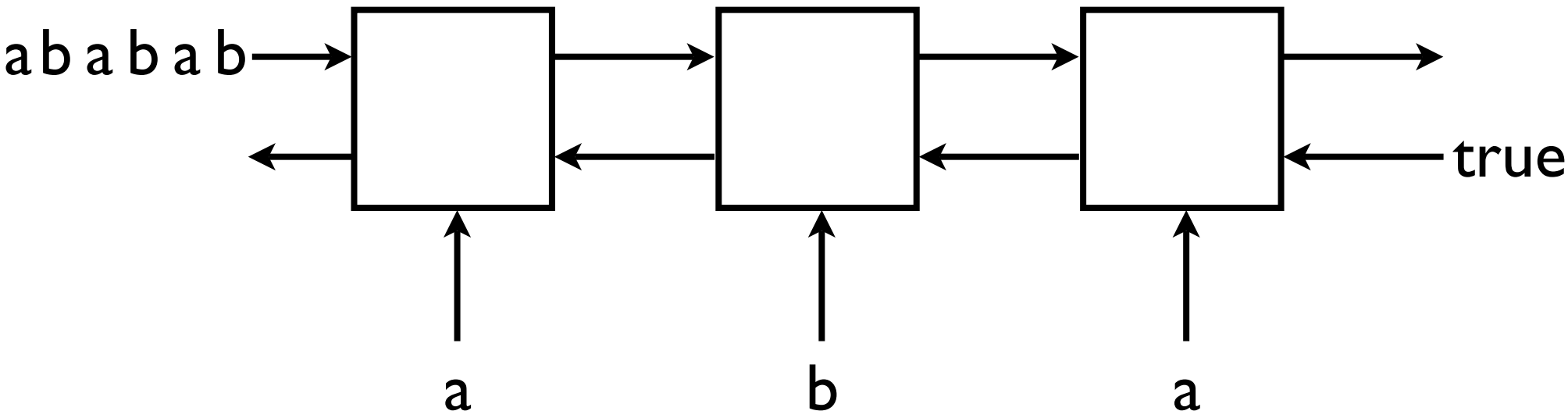
Parallele Algorithmen

Teilworterkennung - Prozessornetz



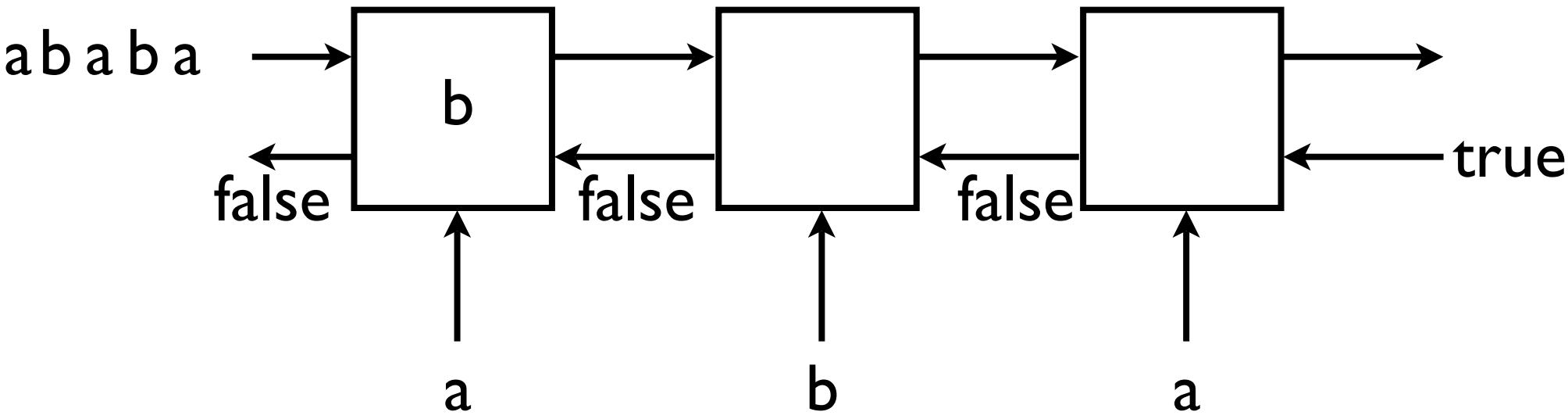
Parallele Algorithmen

Teilworterkennung - Prozessornetz



Parallele Algorithmen

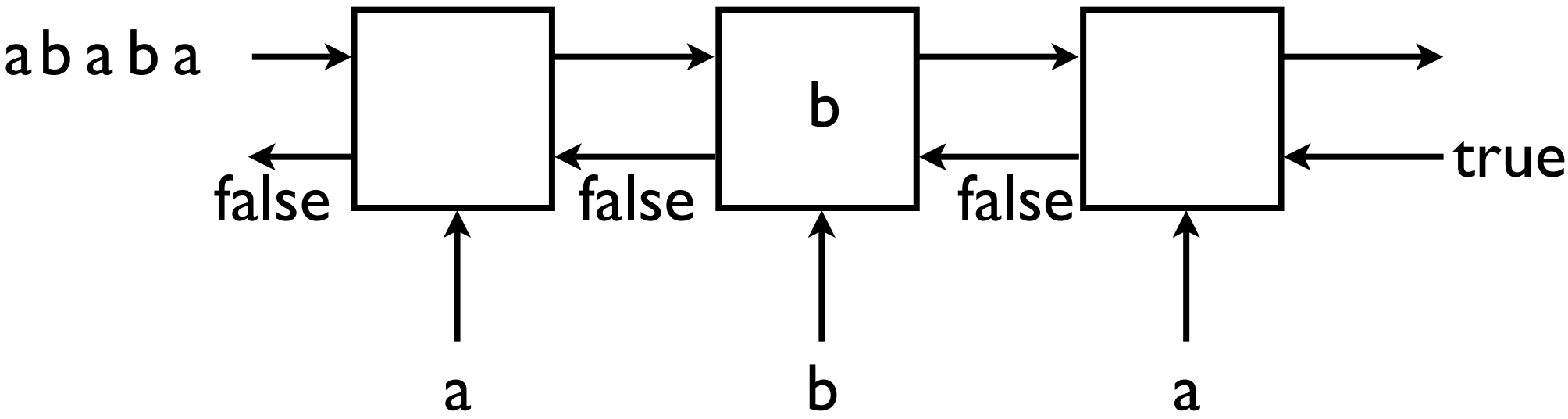
Teilworterkennung - Prozessornetz



1. Takt

Parallele Algorithmen

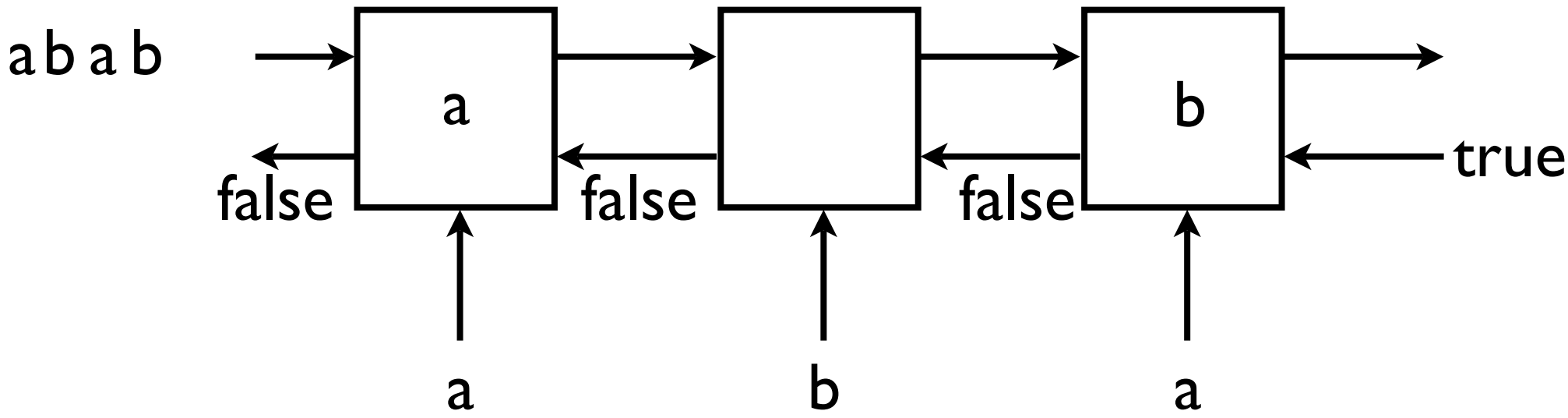
Teilworterkennung - Prozessornetz



2. Takt

Parallele Algorithmen

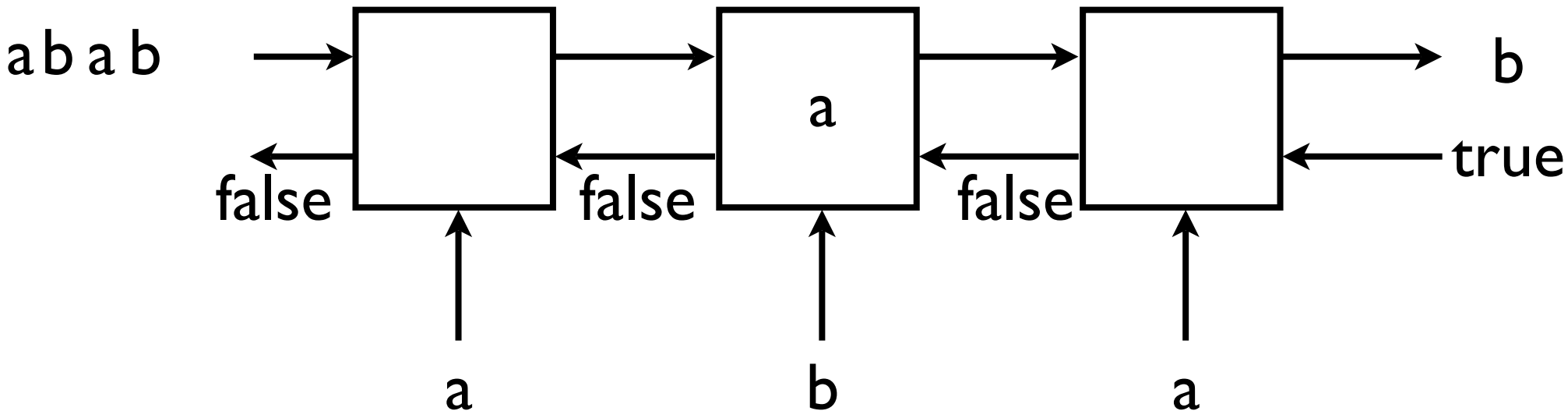
Teilworterkennung - Prozessornetz



3. Takt

Parallele Algorithmen

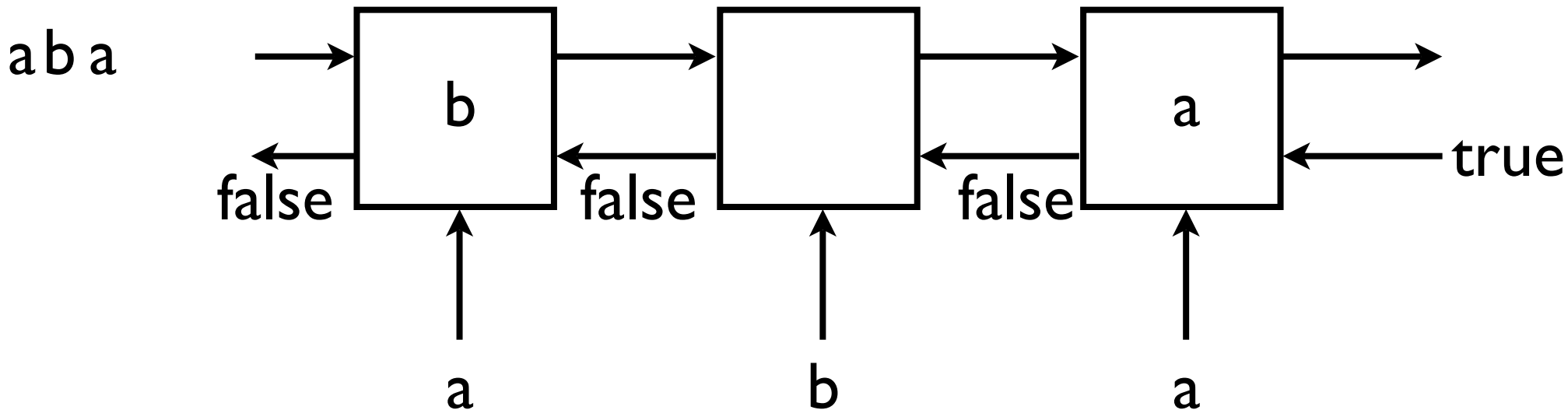
Teilworterkennung - Prozessornetz



4. Takt

Parallele Algorithmen

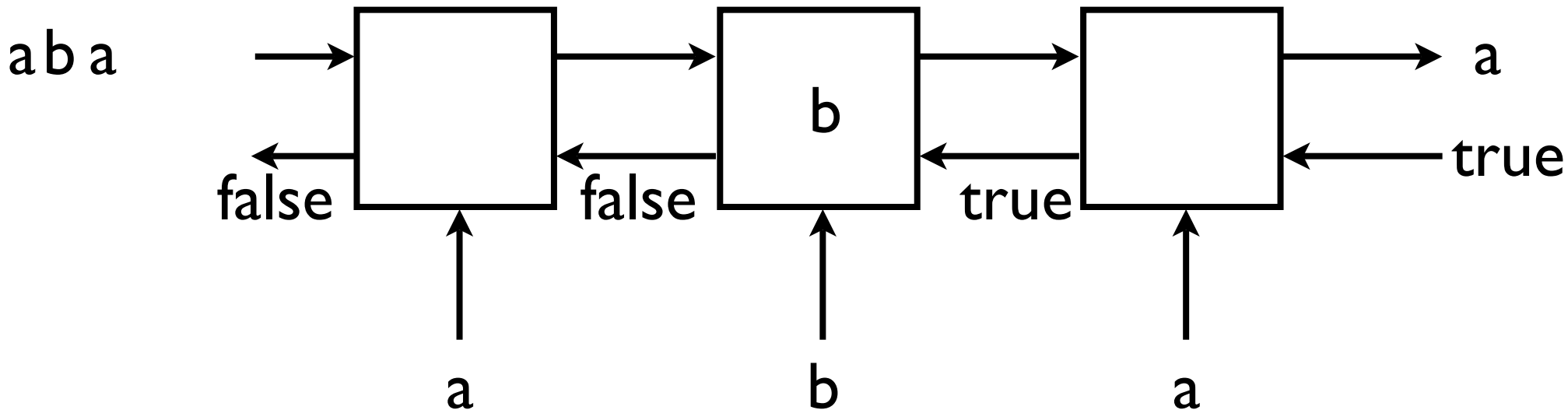
Teilworterkennung - Prozessornetz



5. Takt

Parallele Algorithmen

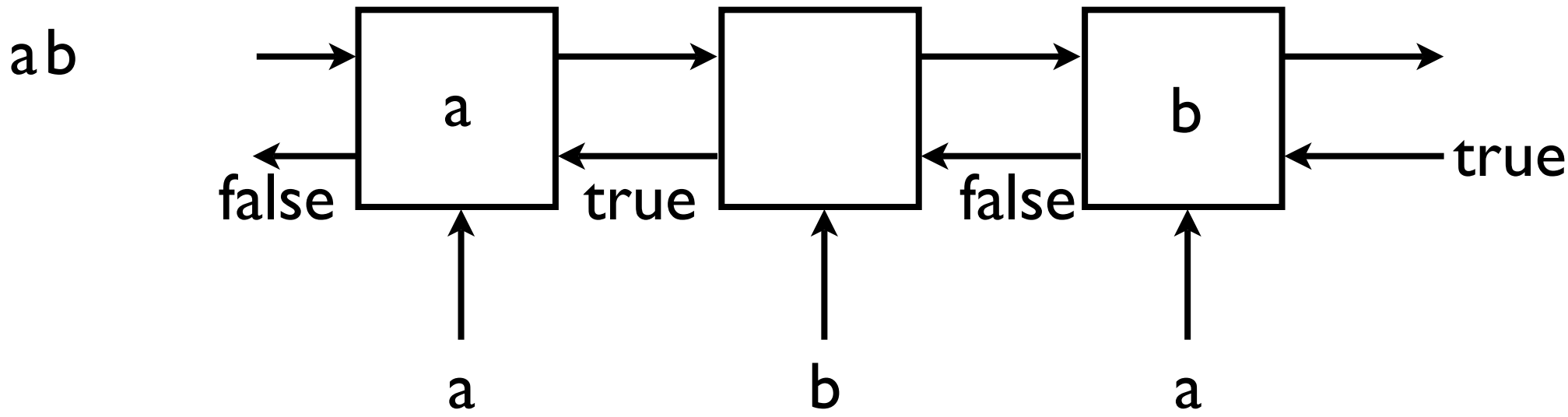
Teilworterkennung - Prozessornetz



6. Takt

Parallele Algorithmen

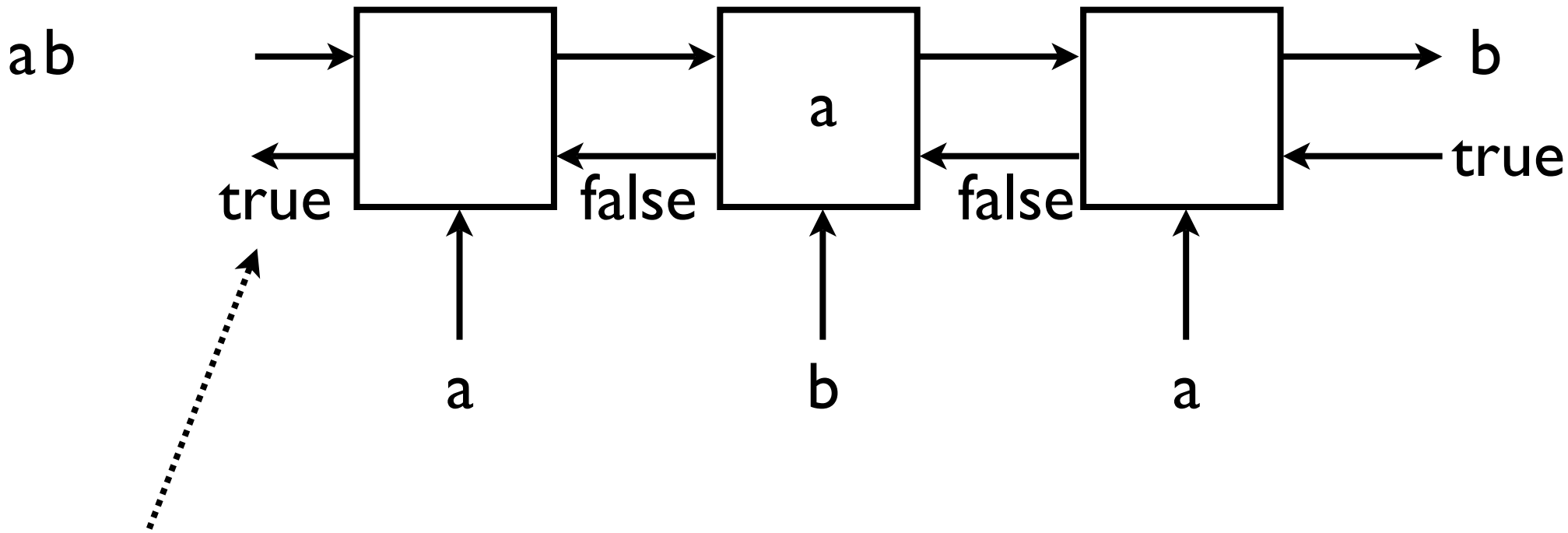
Teilworterkennung - Prozessornetz



7. Takt

Parallele Algorithmen

Teilworterkennung - Prozessornetz



erstmalig aba
in Eingabe erkannt

8. Takt

Parallele Algorithmen
Teilworterkennung - Effizienzvergleich

- Sequentieller Fall mit einem Prozessor:

$$T_1(n)=3n$$

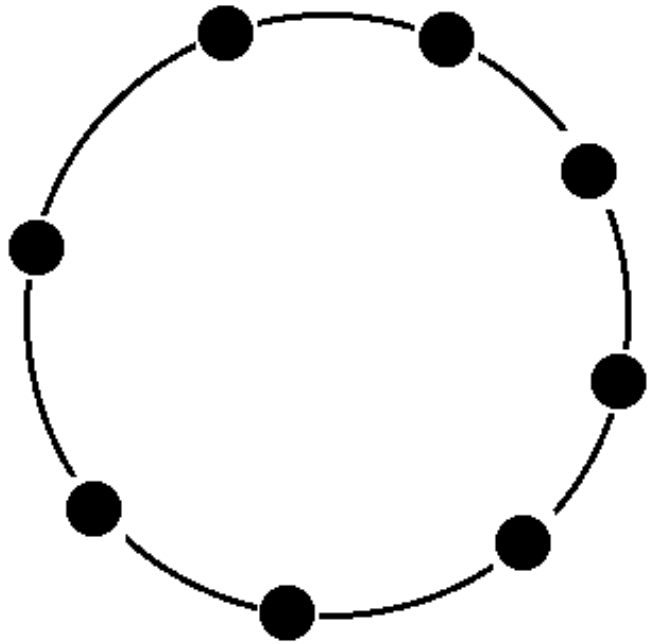
- Paralleler Fall mit drei Prozessoren:

$$T_3(n)=2n-1$$

- Speed-up: $S_3(n)=3n/(2n-1)\approx 3/2$
- Effizienz: $E_3(n)=S_3(n)/3\approx 1/2$
- Teilworterkennung *nicht* optimal.

Leader election in Netzen
Motivation

- automatischer Neustart eines Rechnernetzes
- Bestimmung eines ausgezeichneten Rechners für erste Verwaltungaufgaben
- Bestimmen eines neuen Leaders nach Ausfall des alten
- Erzeugung eines Token in Tokennetzwerken



Varianten:
- Uniformität
- Anonymität
- Synchronität
- Direktionalität von Kanten
- Netztopologien (hier nur Ringe)

Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
Motivation

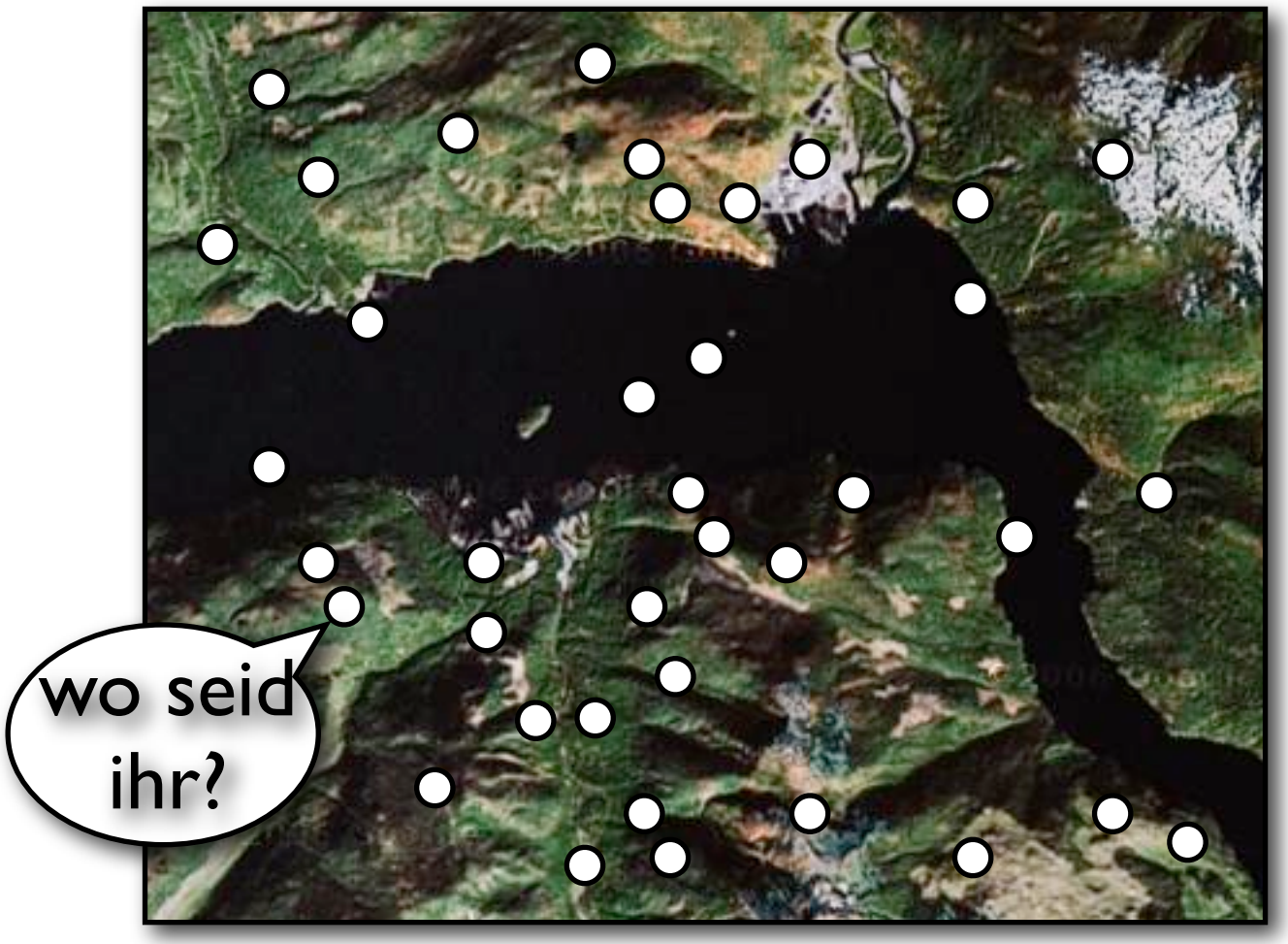
- Sensornetze

Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
Motivation

- Sensornetze



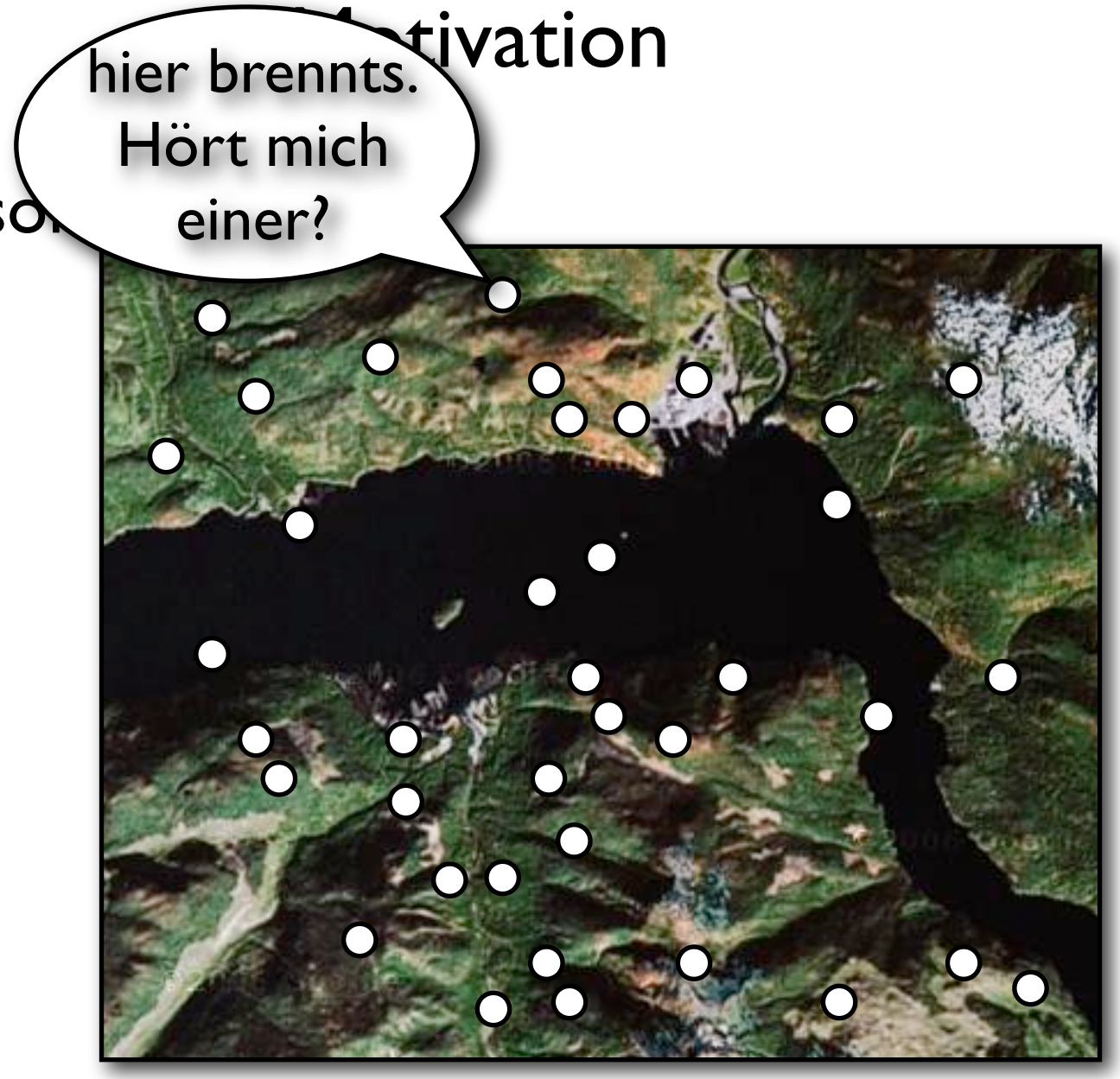
Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen

Motivation

- Sensor

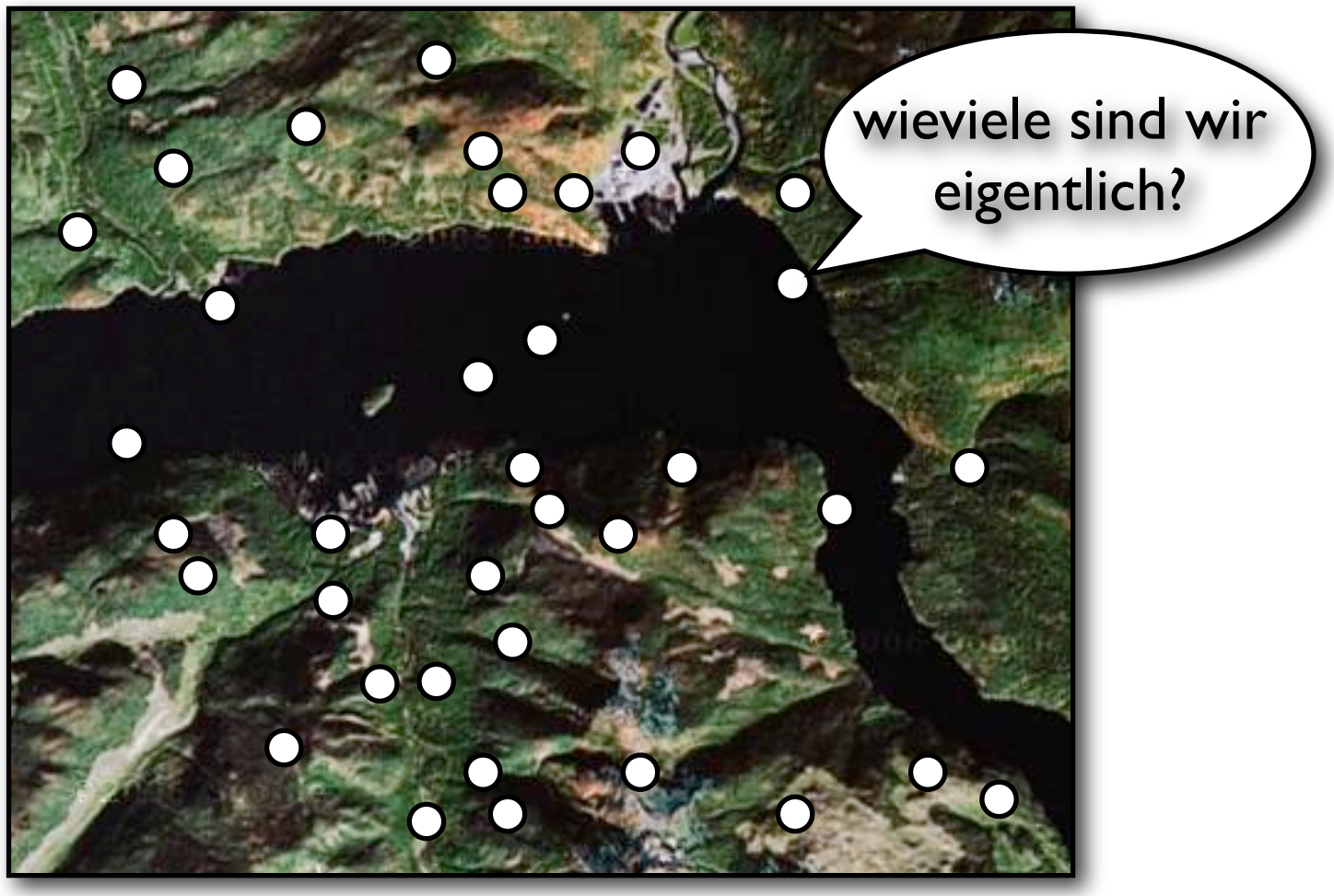


Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
Motivation

- Sensornetze



Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
Motivation

- Sensornetze

Leader election in Netzen
im Ring

Algorithmus [Franklin 1982]:

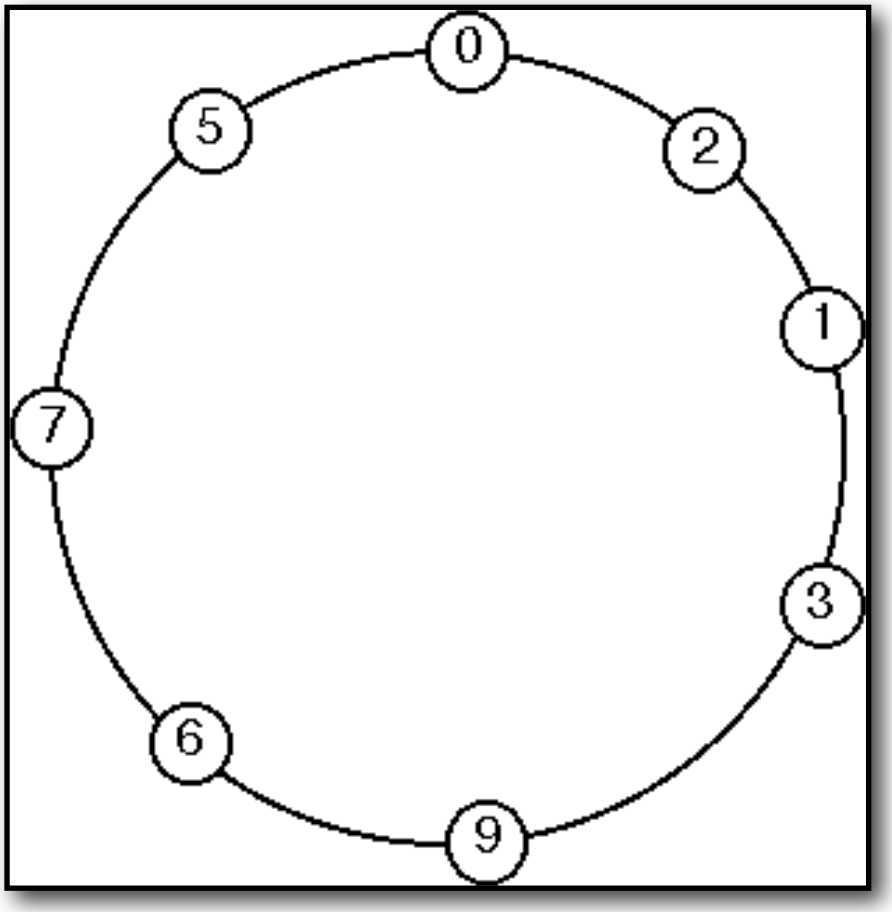
- je Rechner eindeutige Identifikation (id)
- anfangs alle Rechner aktiv (=weiß, schwarz=passiv).

Programm für jeden Rechner:

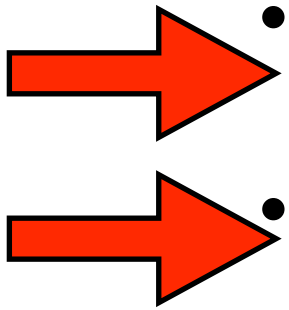
- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader.

Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader

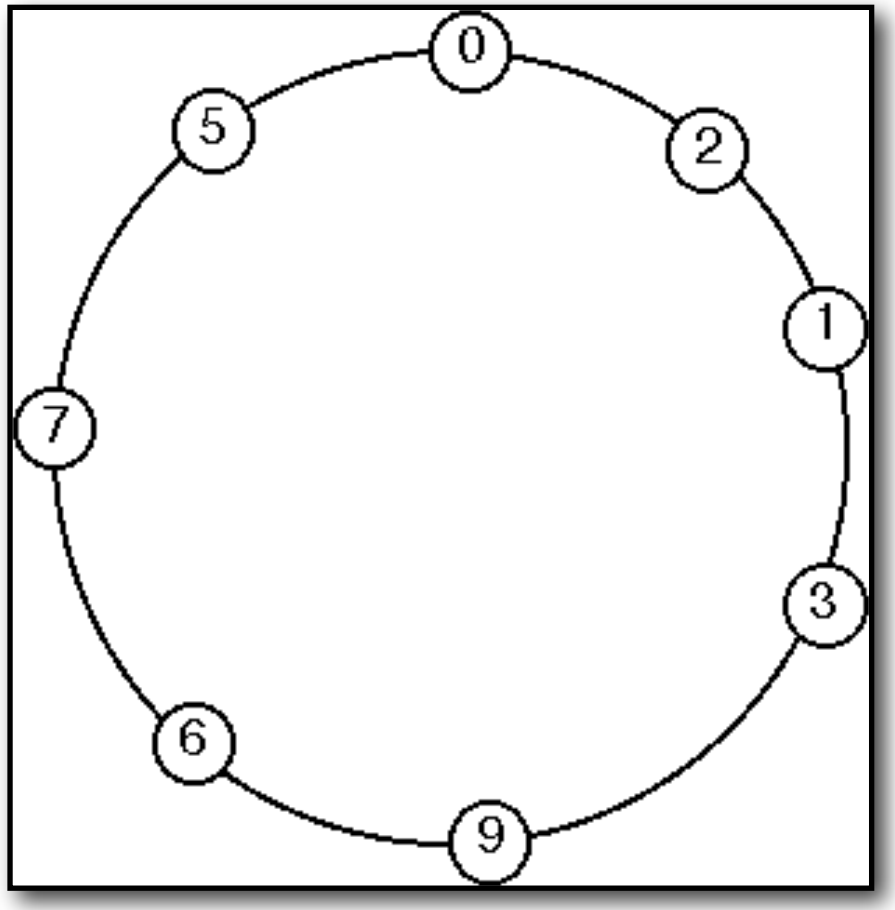


Leader election in Netzen
im Ring - Beispiel

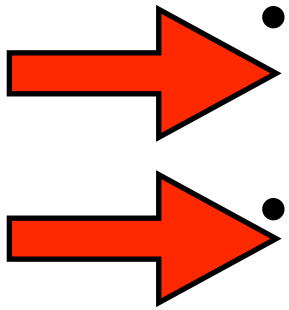


- sende beiden Nachbarn deine Information id

- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader

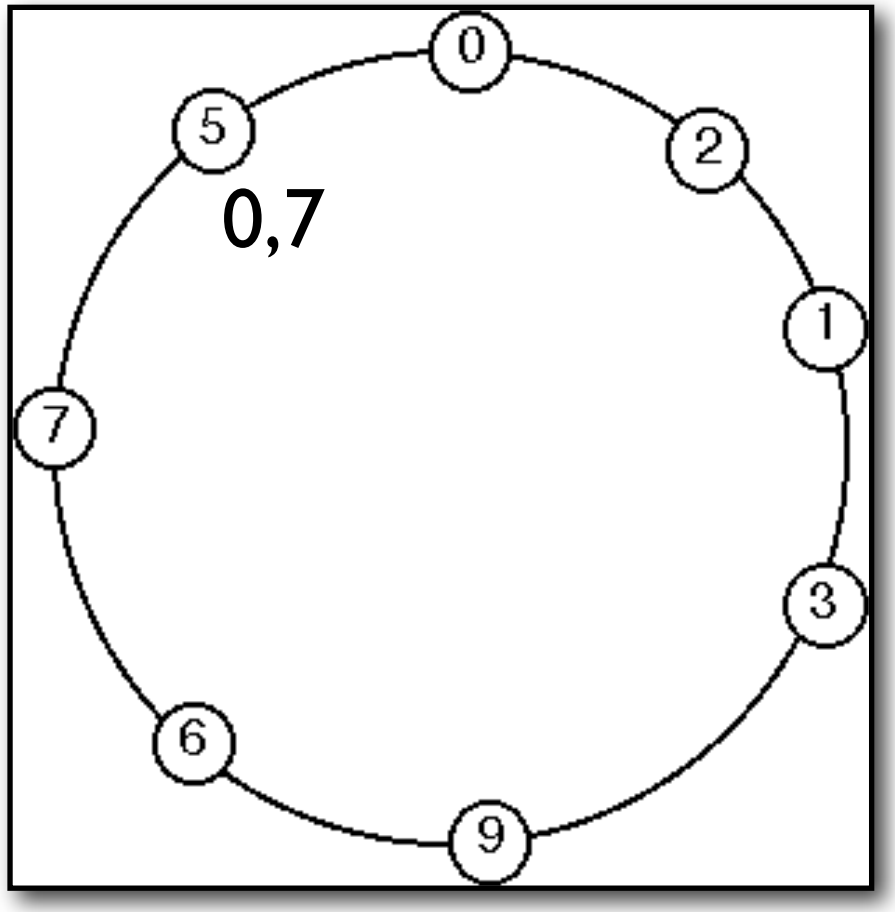


Leader election in Netzen
im Ring - Beispiel

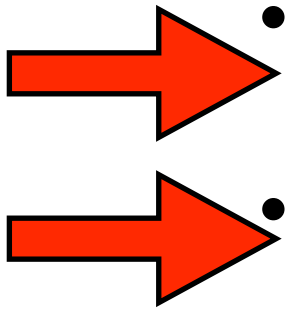


- sende beiden Nachbarn deine Information id

- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader

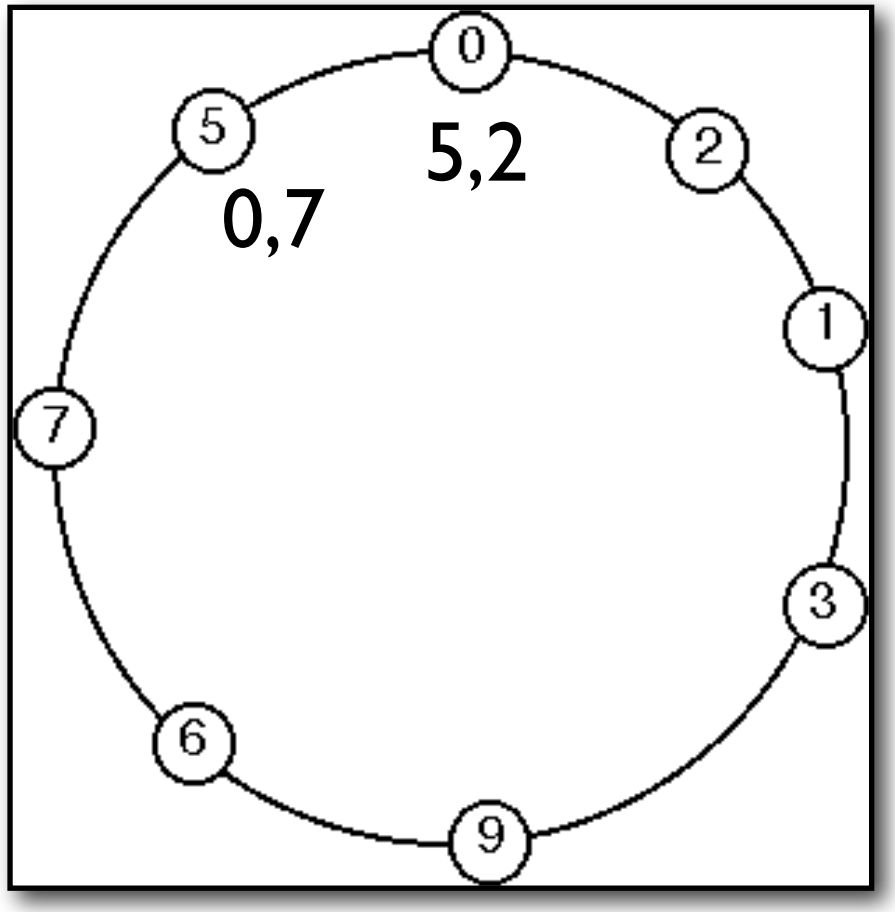


Leader election in Netzen
im Ring - Beispiel

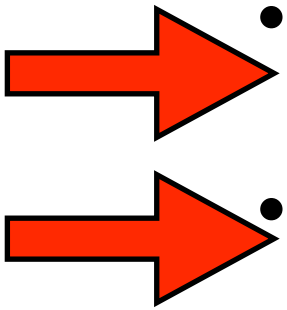


- sende beiden Nachbarn deine Information id

- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader

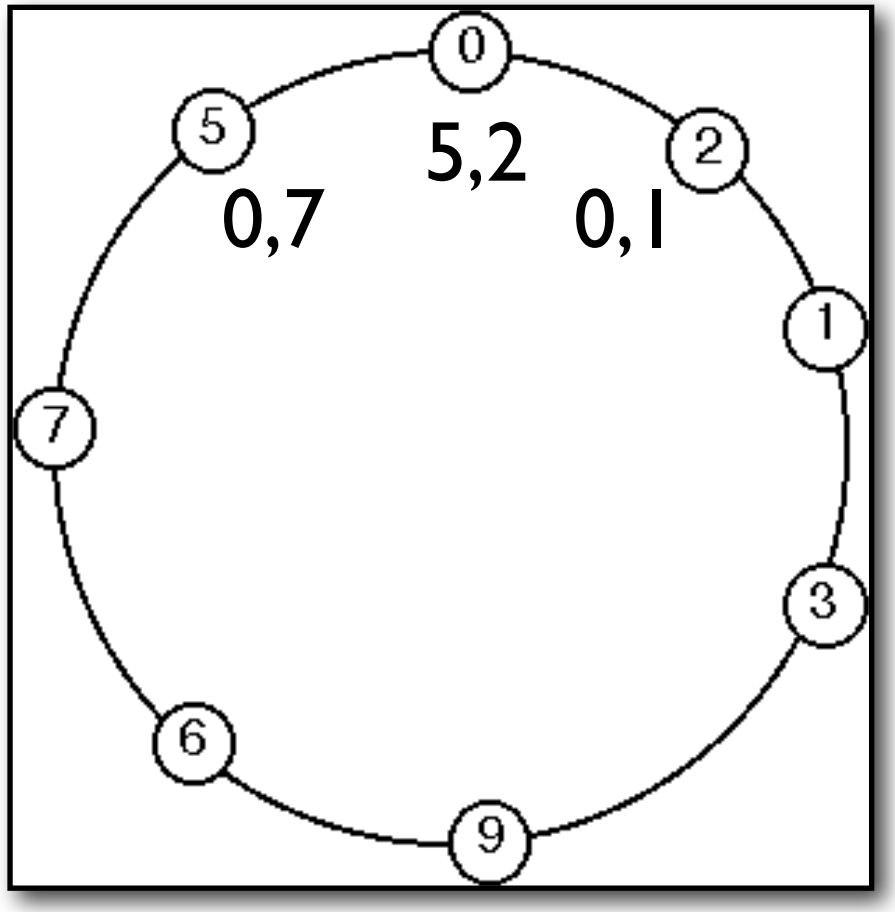


Leader election in Netzen
im Ring - Beispiel

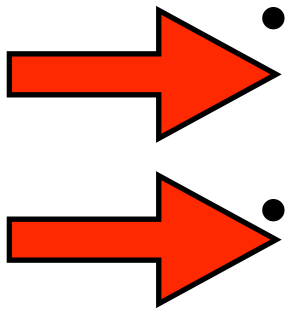


- sende beiden Nachbarn deine Information id

- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader

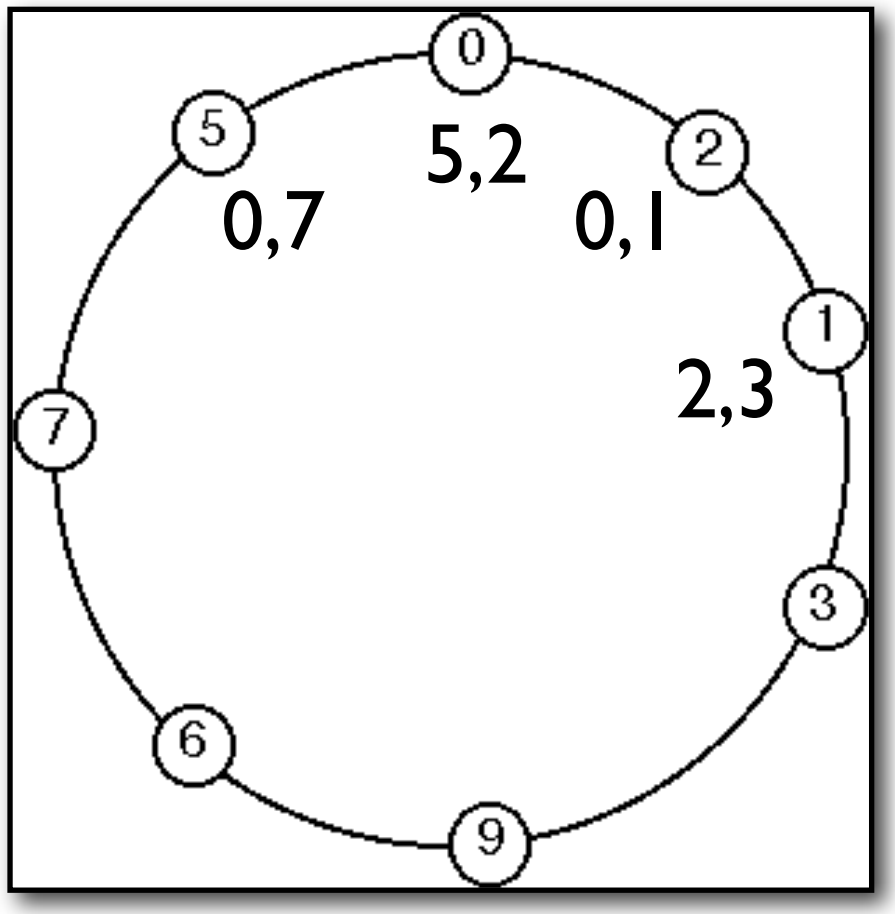


Leader election in Netzen
im Ring - Beispiel

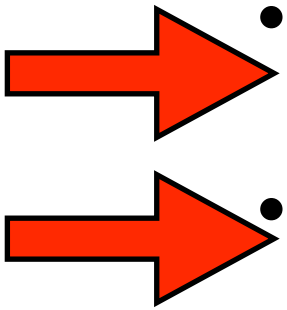


- sende beiden Nachbarn deine Information id

- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader

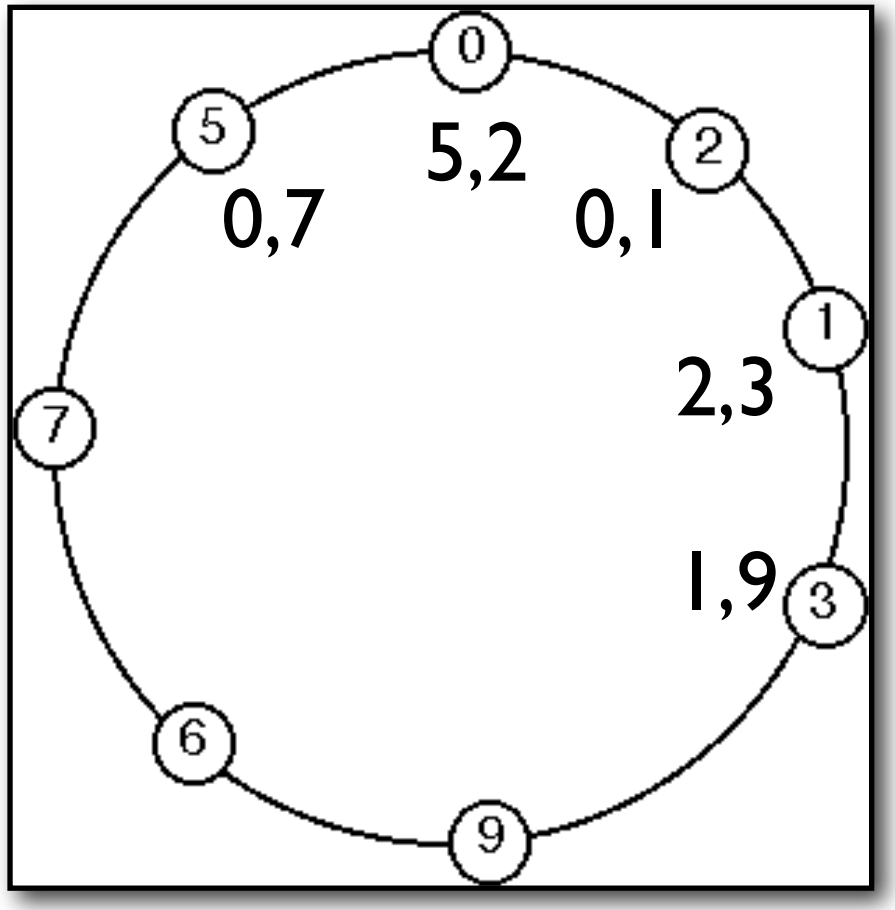


Leader election in Netzen
im Ring - Beispiel

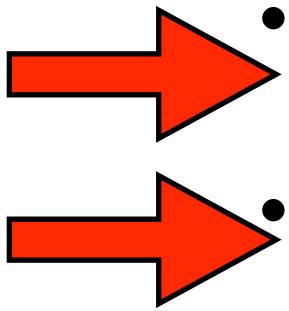


- sende beiden Nachbarn deine Information id

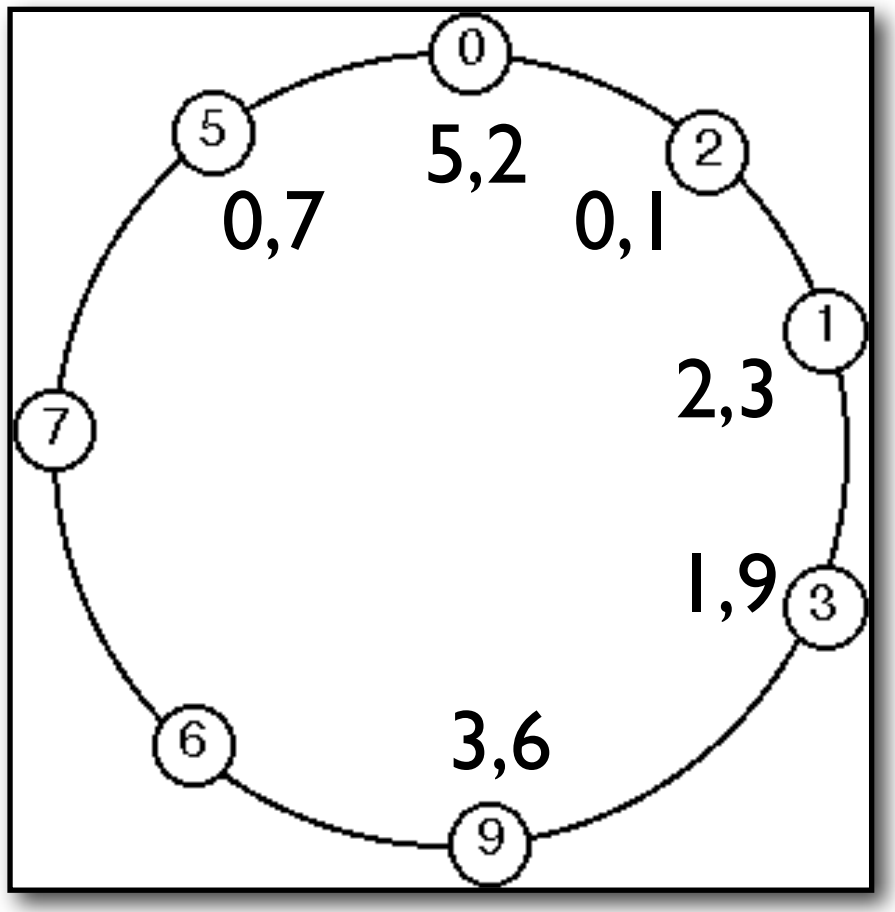
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



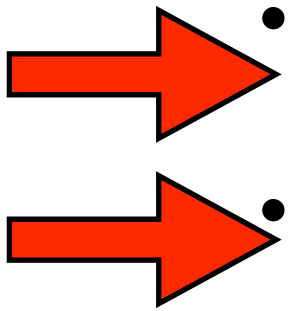
Leader election in Netzen
im Ring - Beispiel



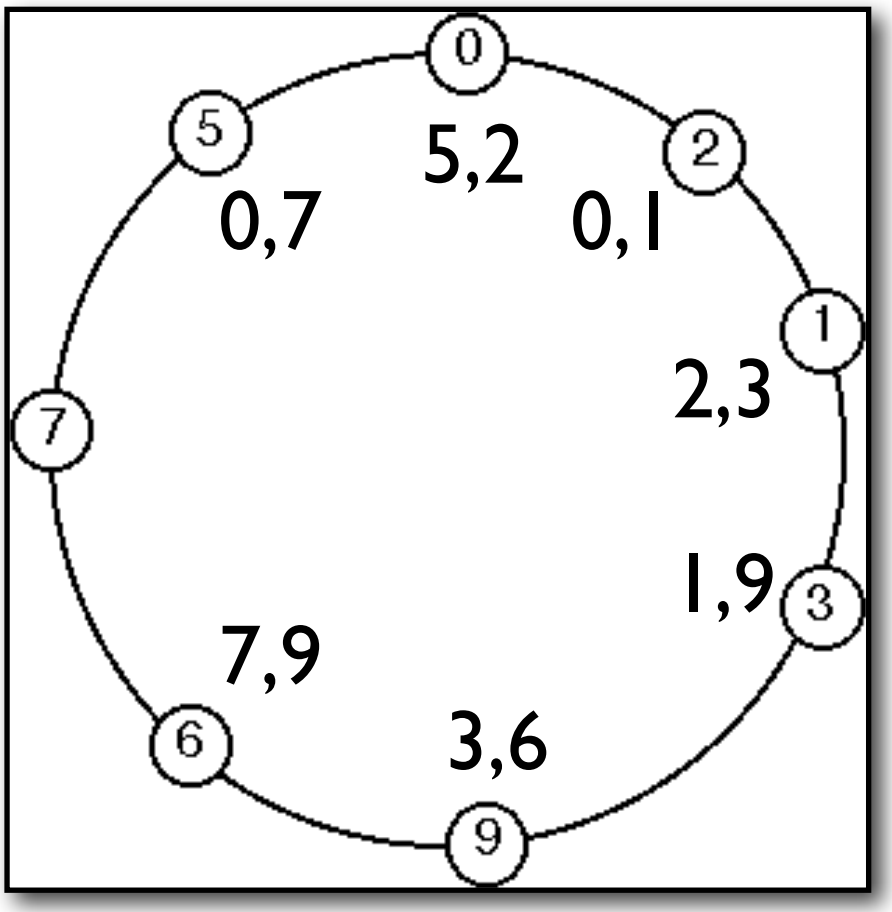
- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



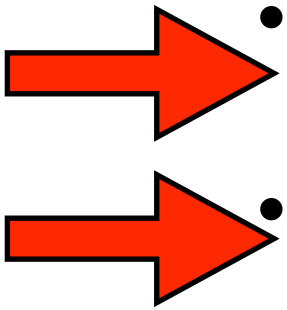
Leader election in Netzen
im Ring - Beispiel



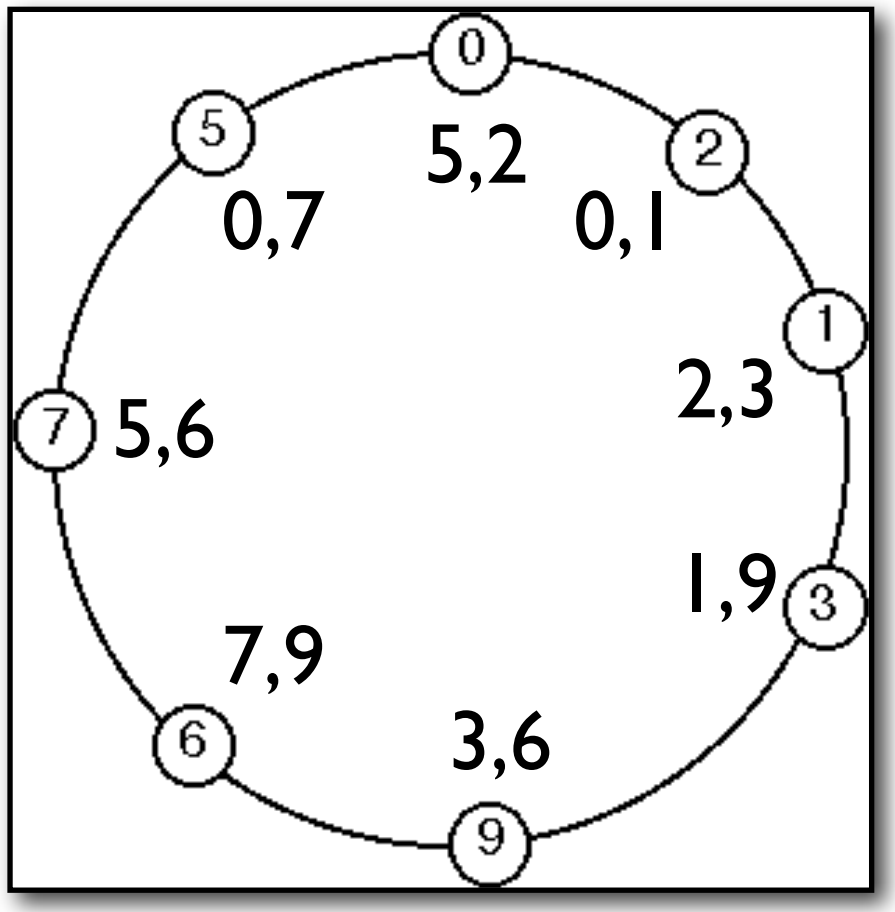
- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



Leader election in Netzen
im Ring - Beispiel

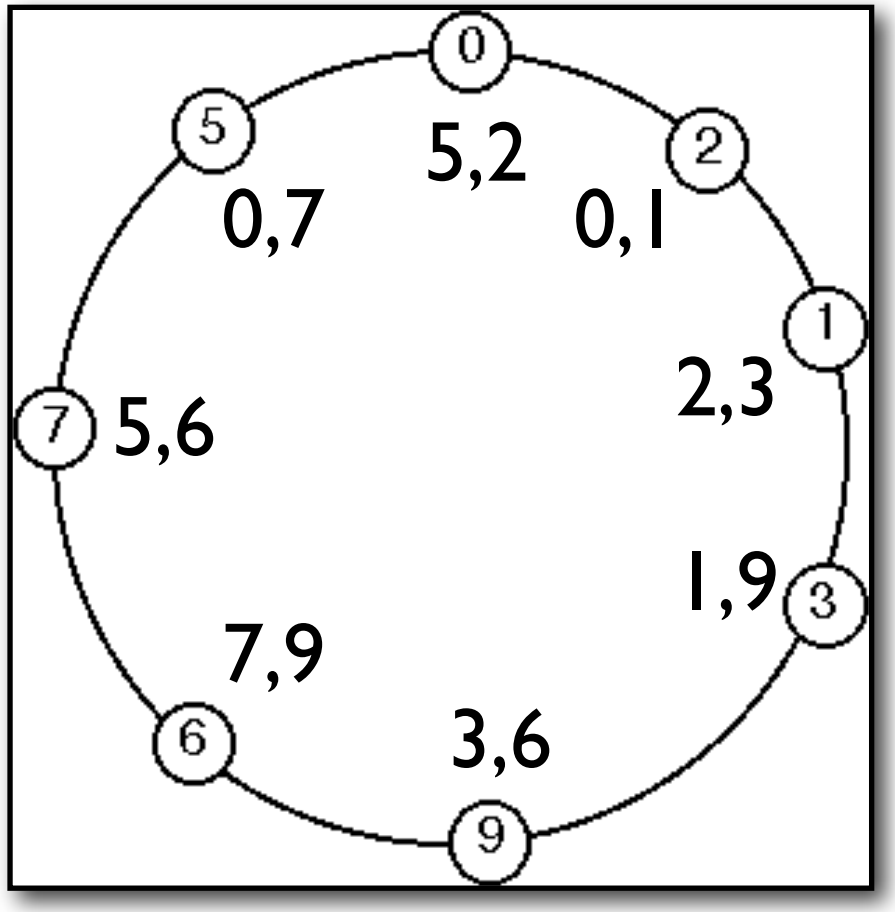


- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



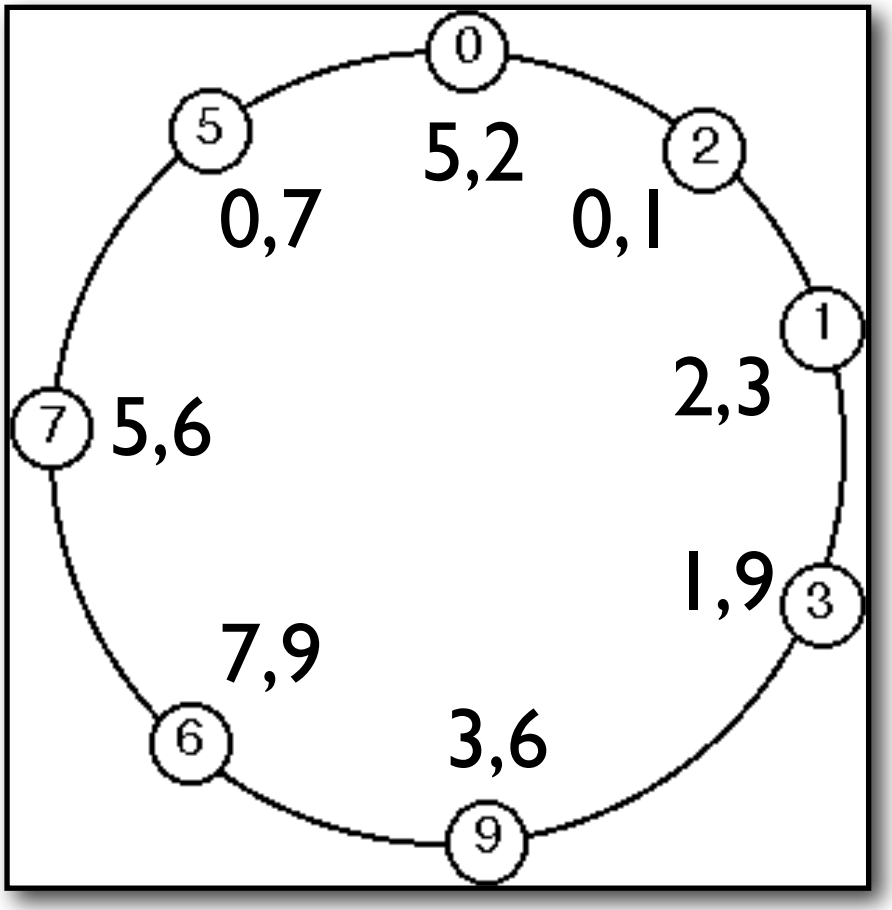
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



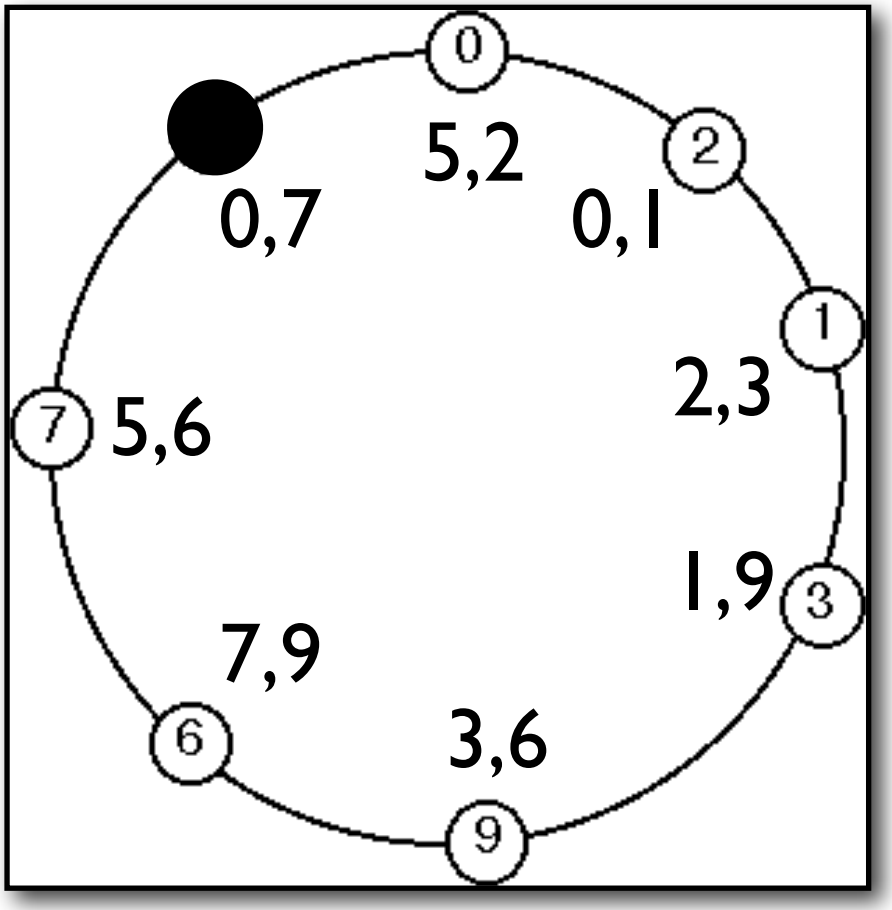
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



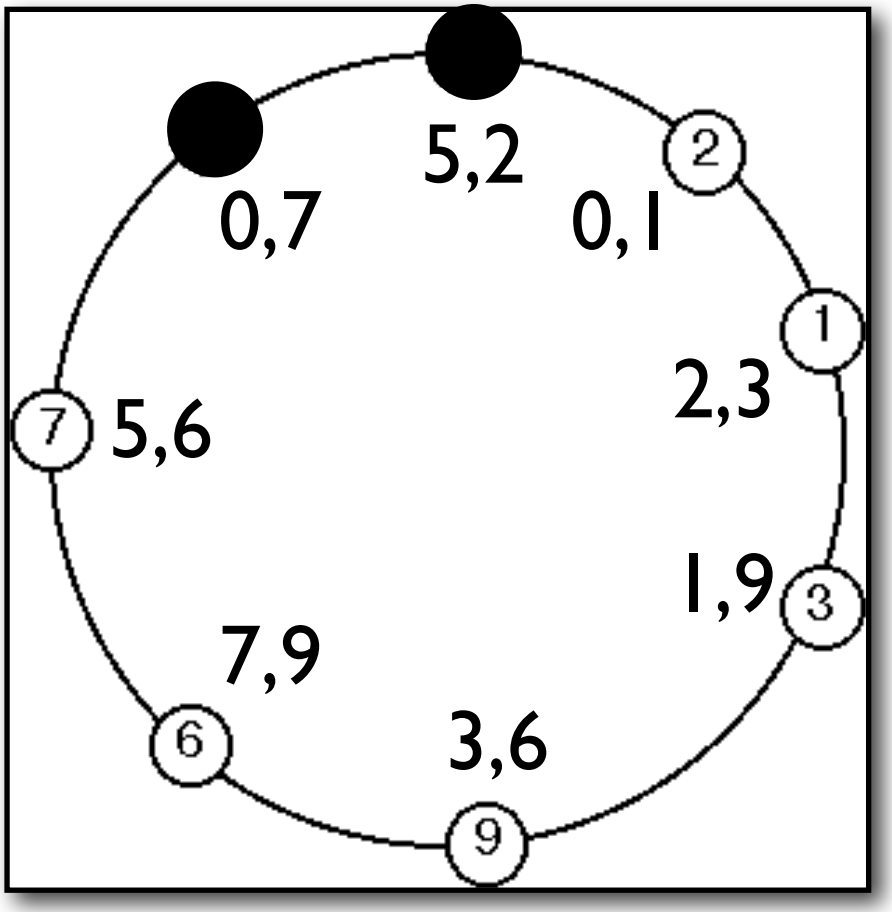
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



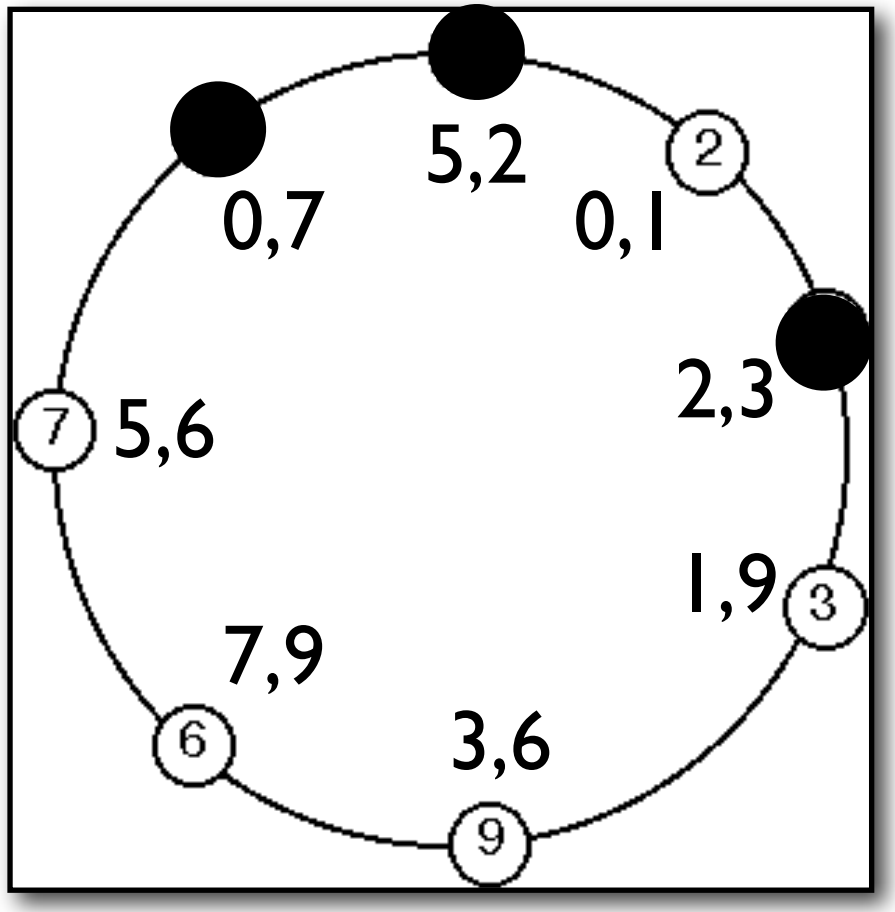
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



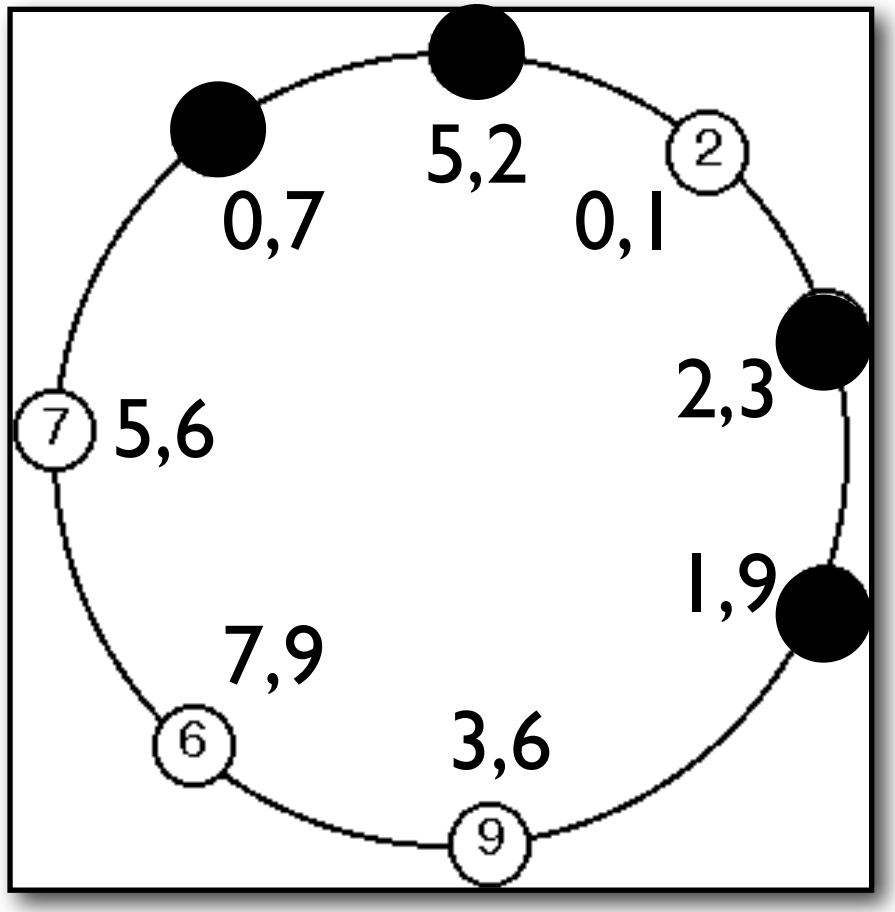
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



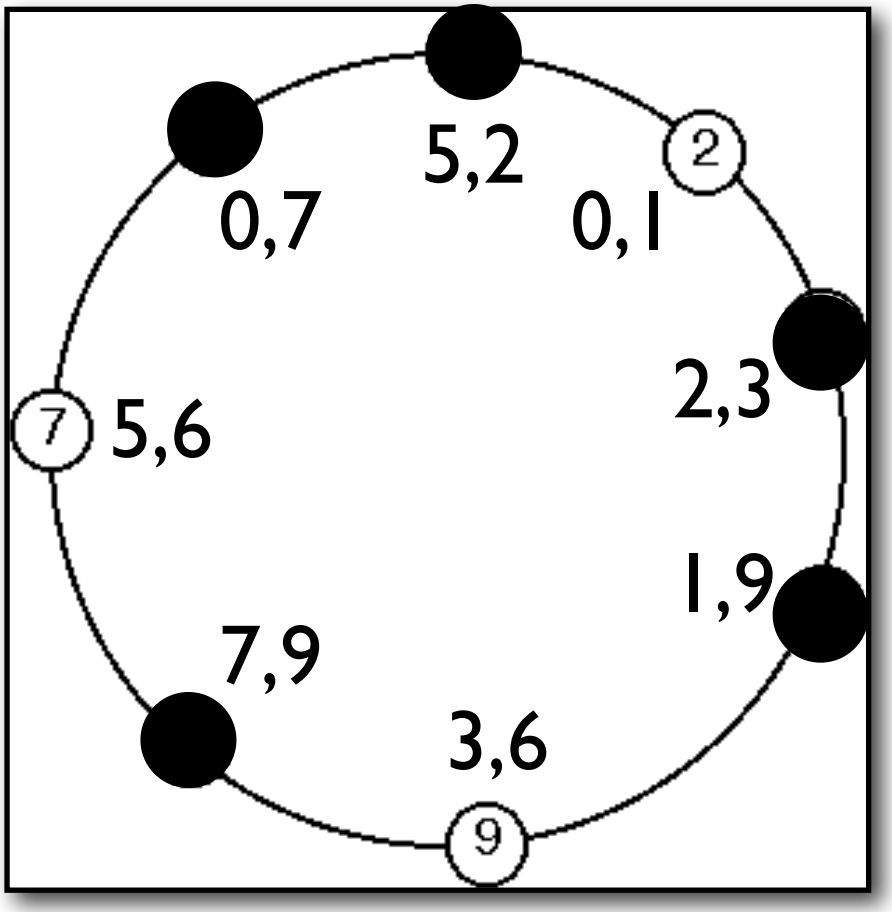
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



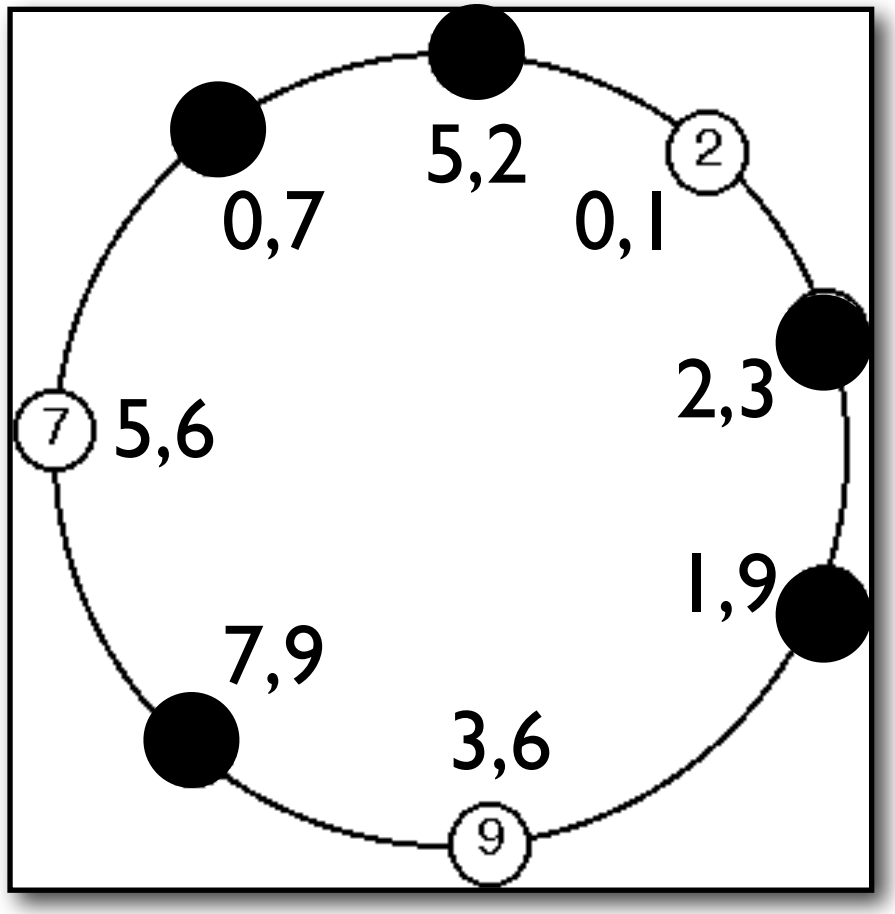
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



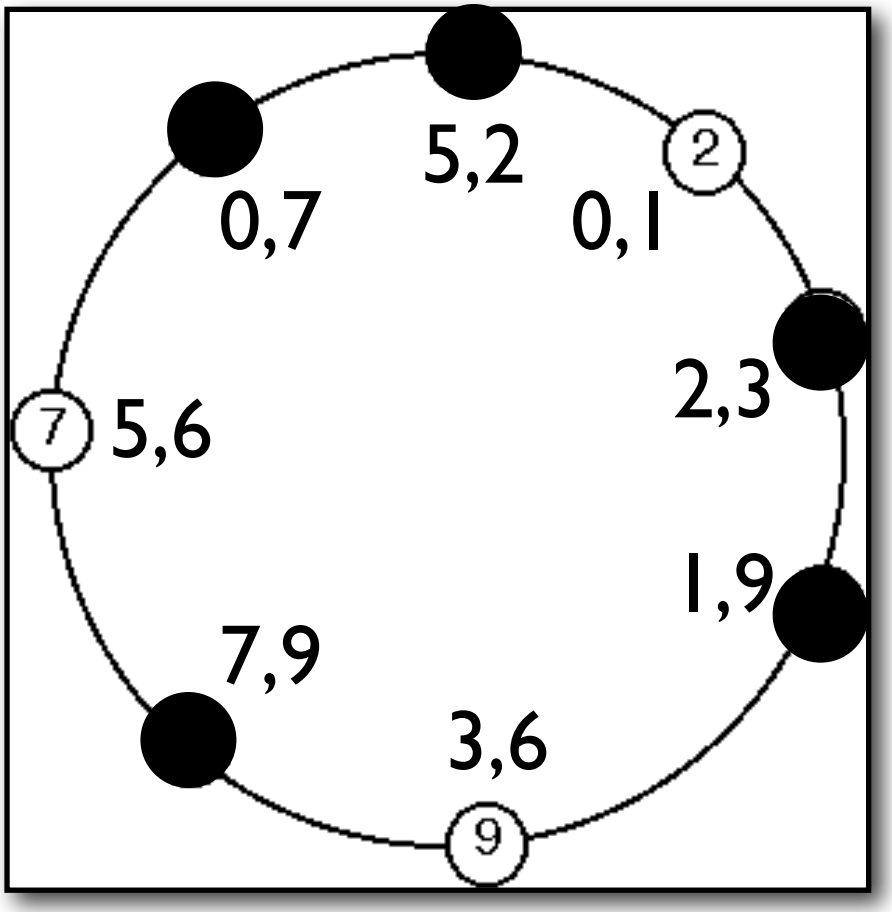
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



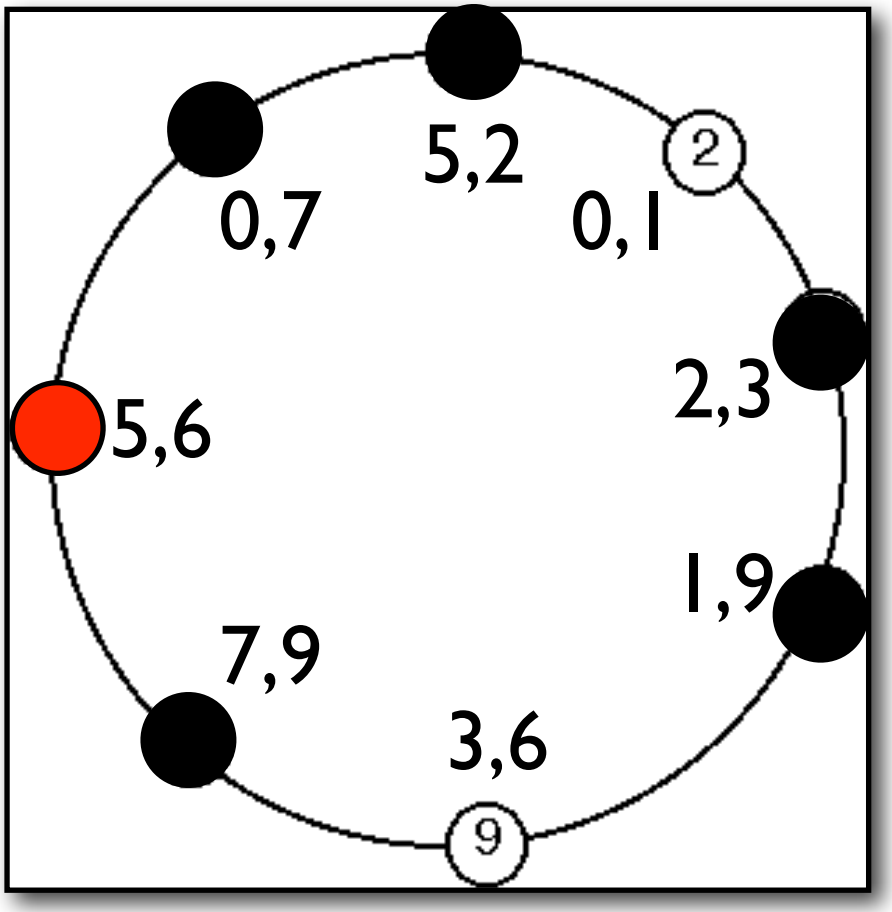
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



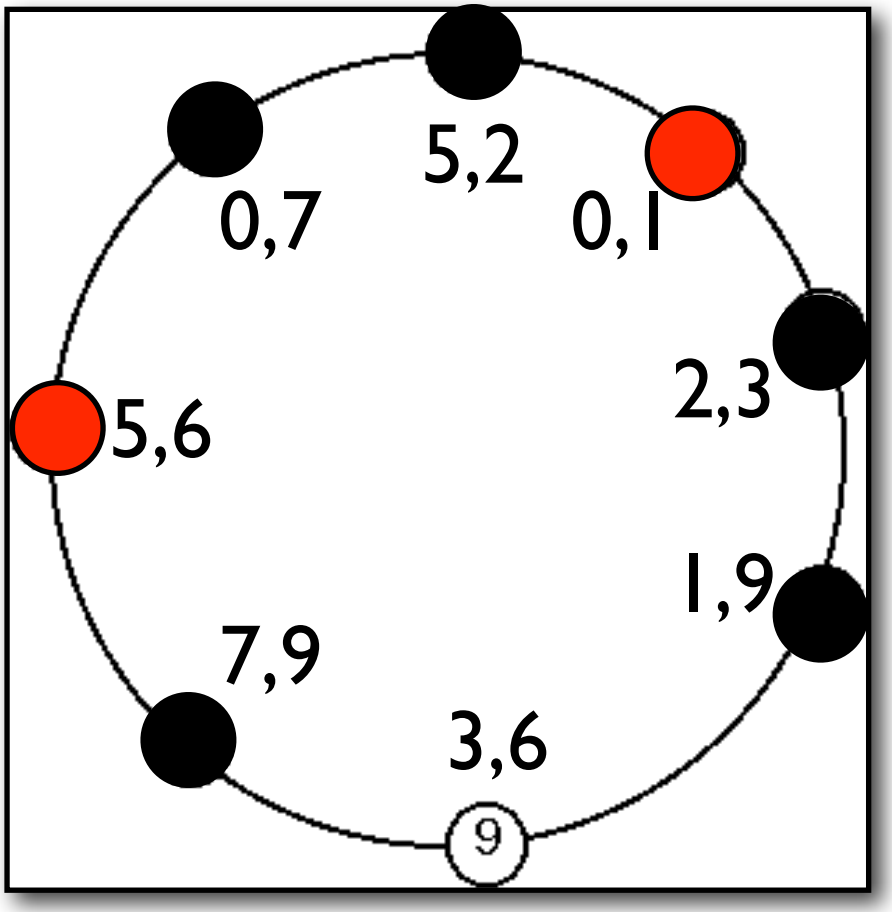
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



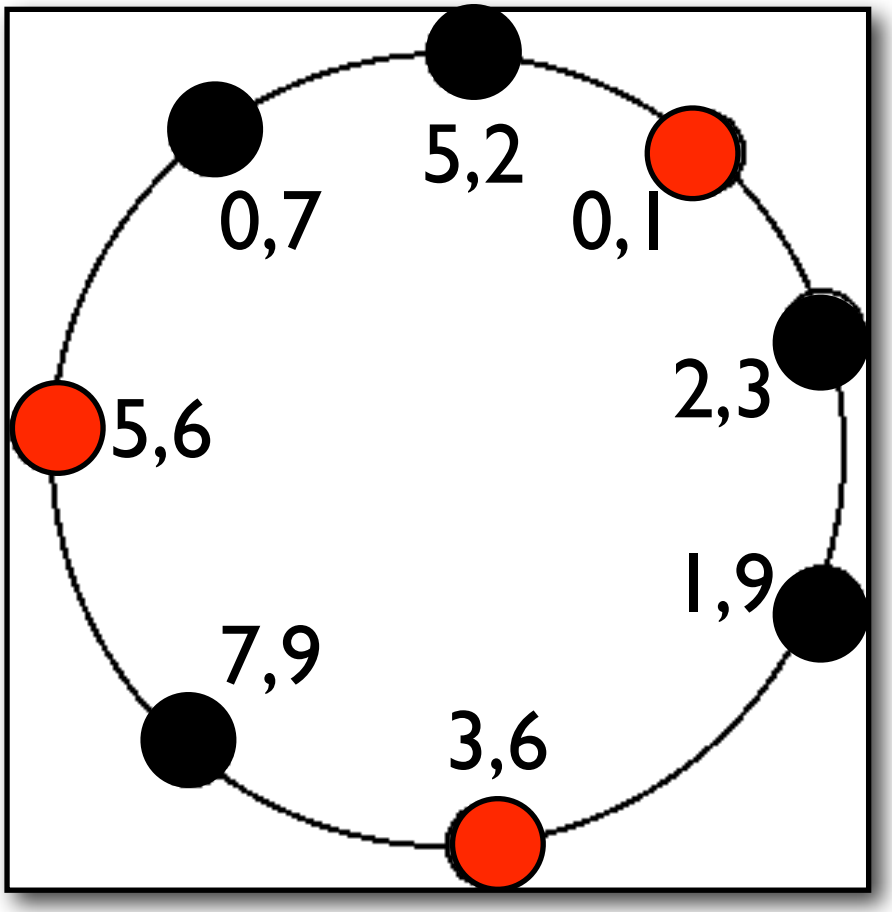
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



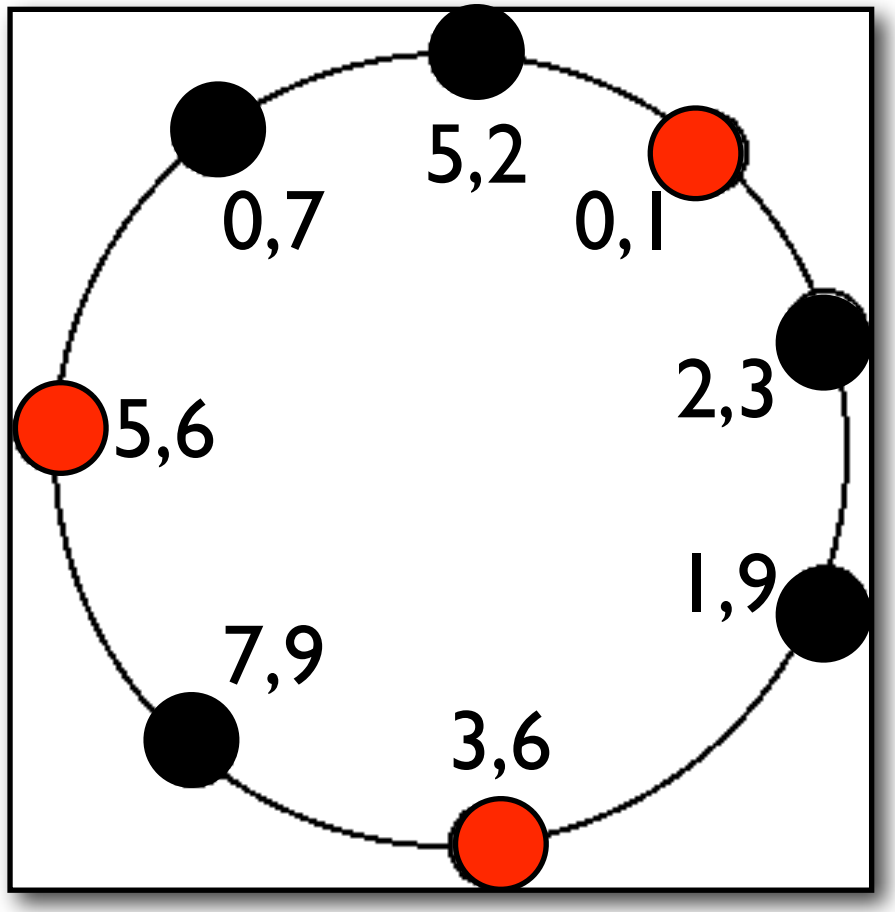
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



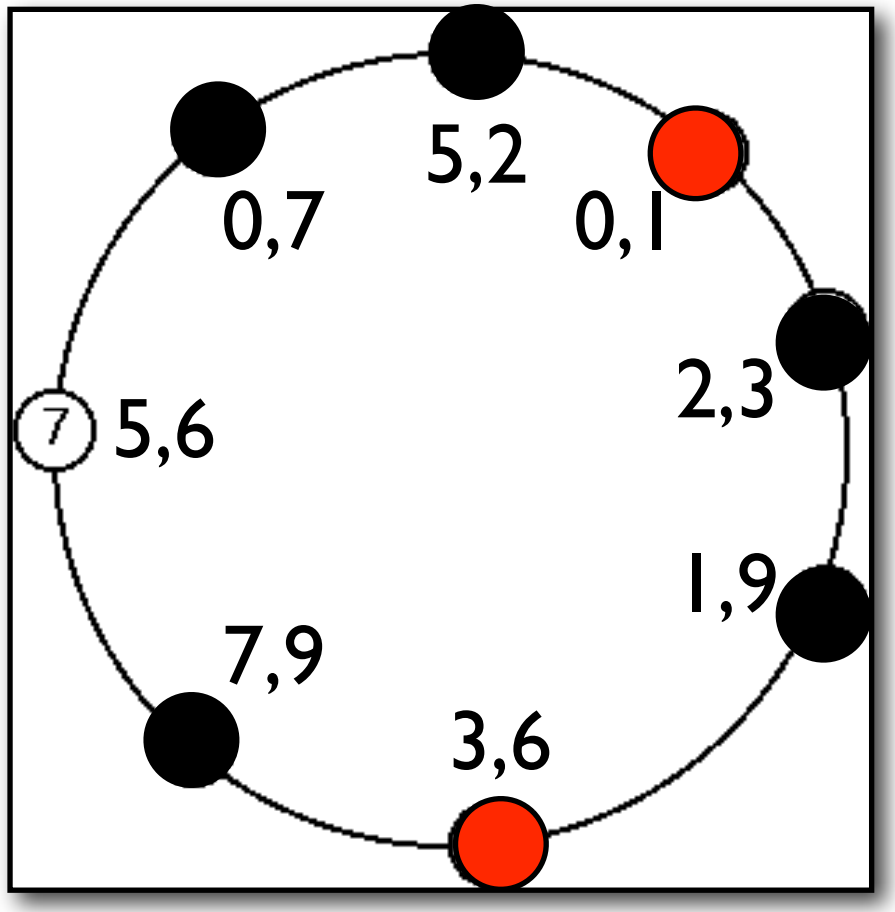
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



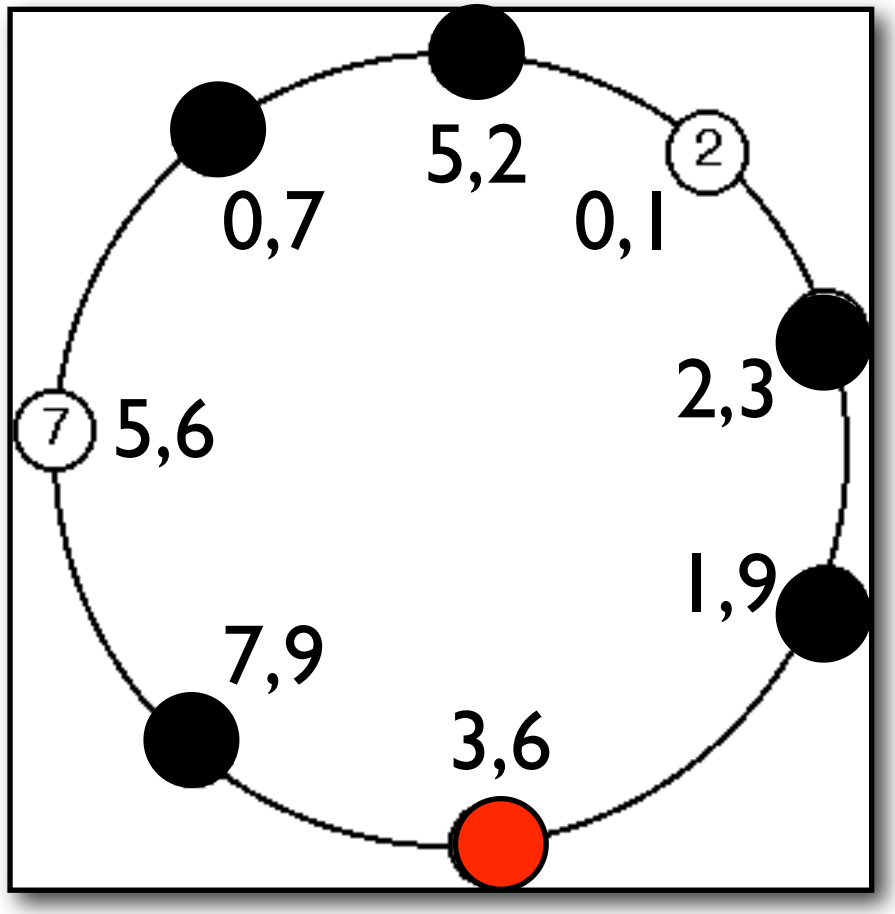
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



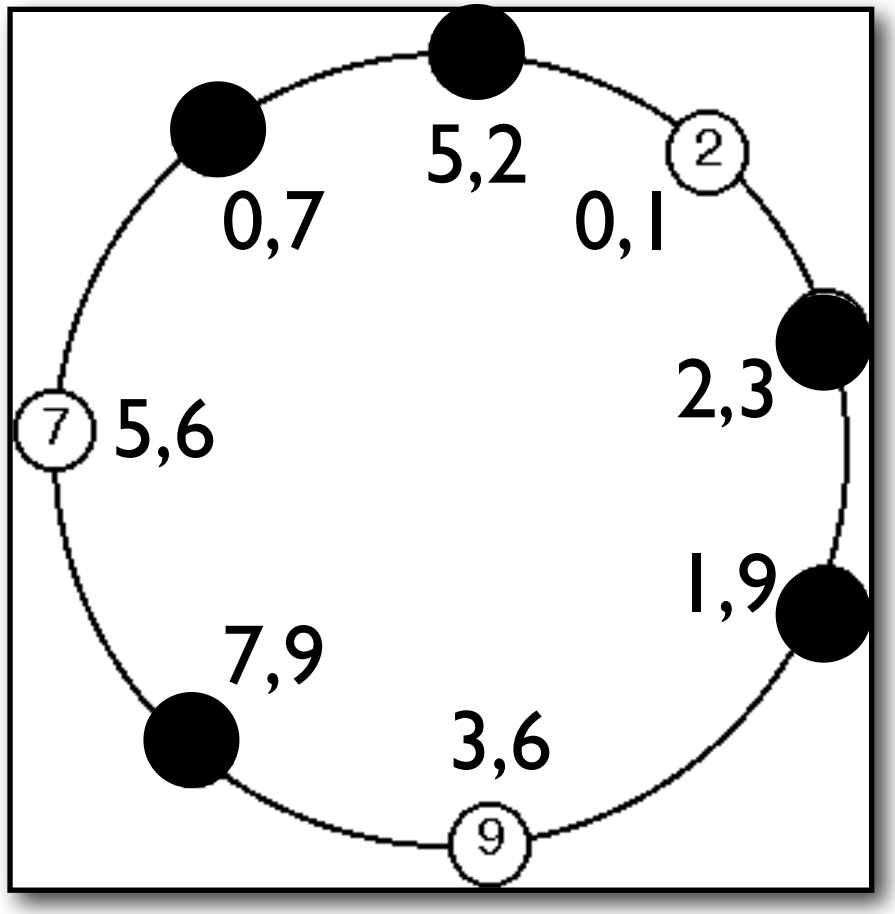
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



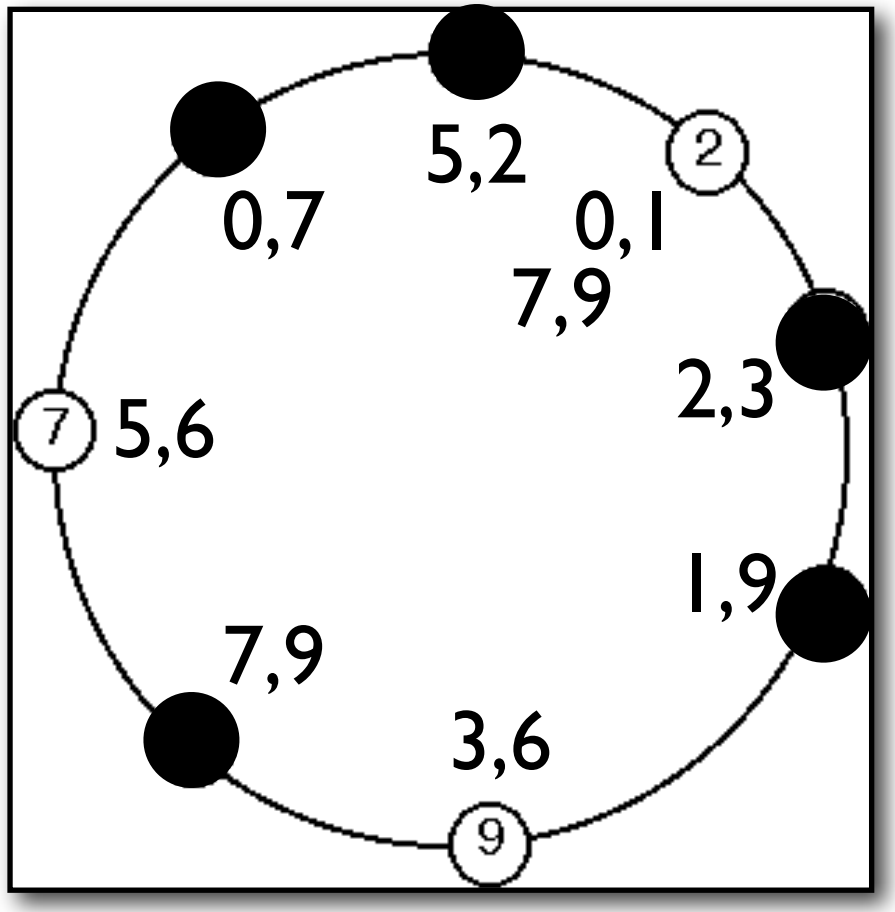
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



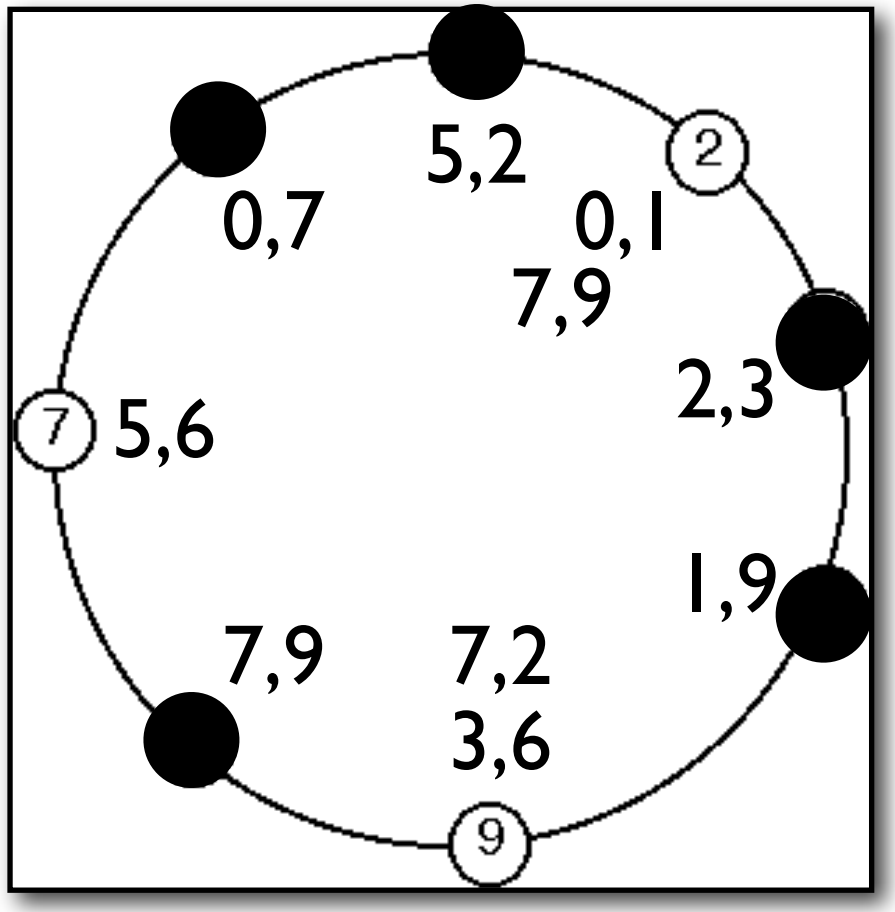
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls id_1>id oder id_2>id dann werde passiv
- falls id_1,id_2<id dann alles nochmal (nächste Runde)
- falls id_1=id_2=id dann erkläre dich zum Leader



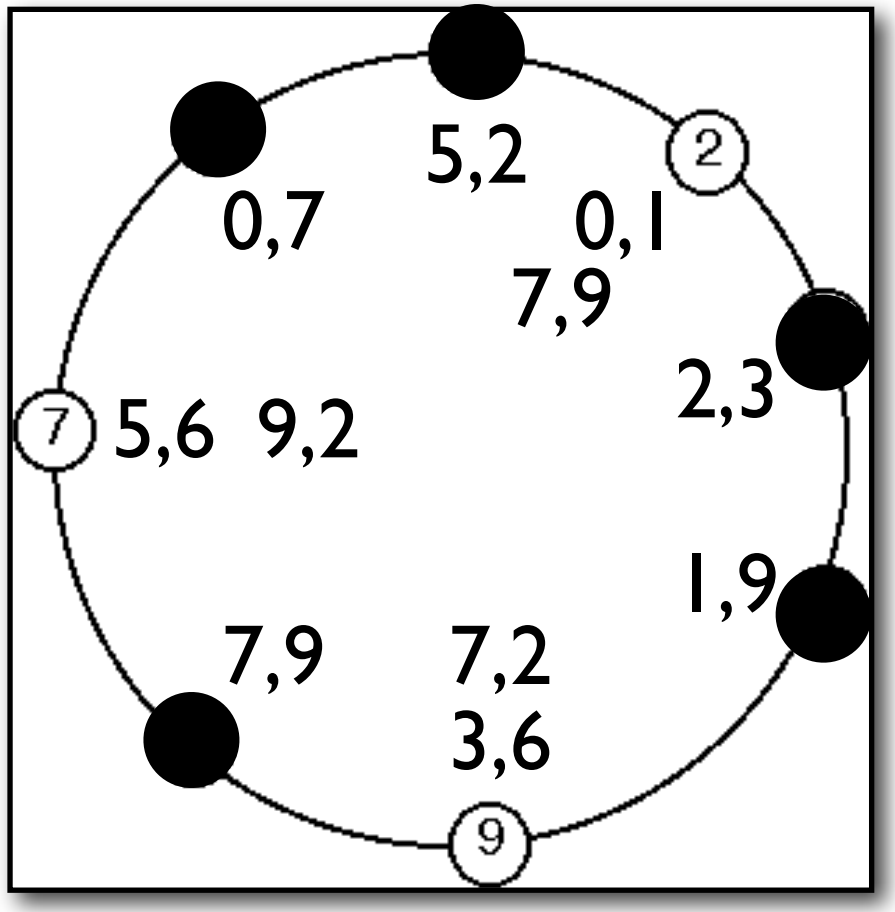
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



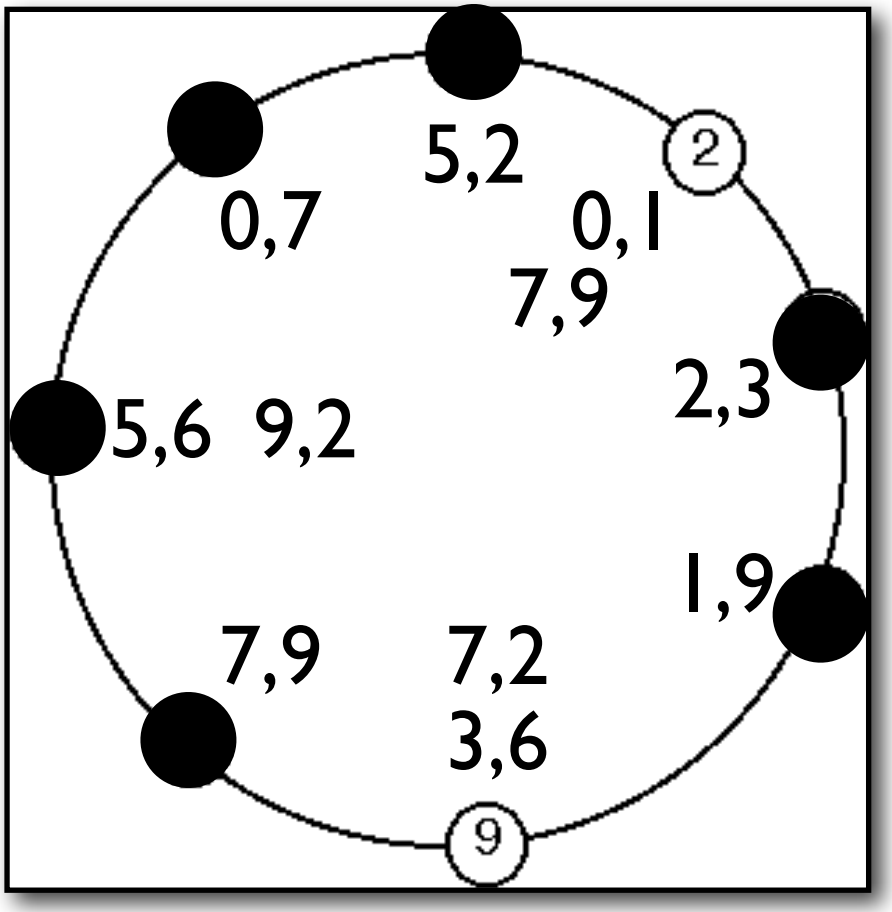
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



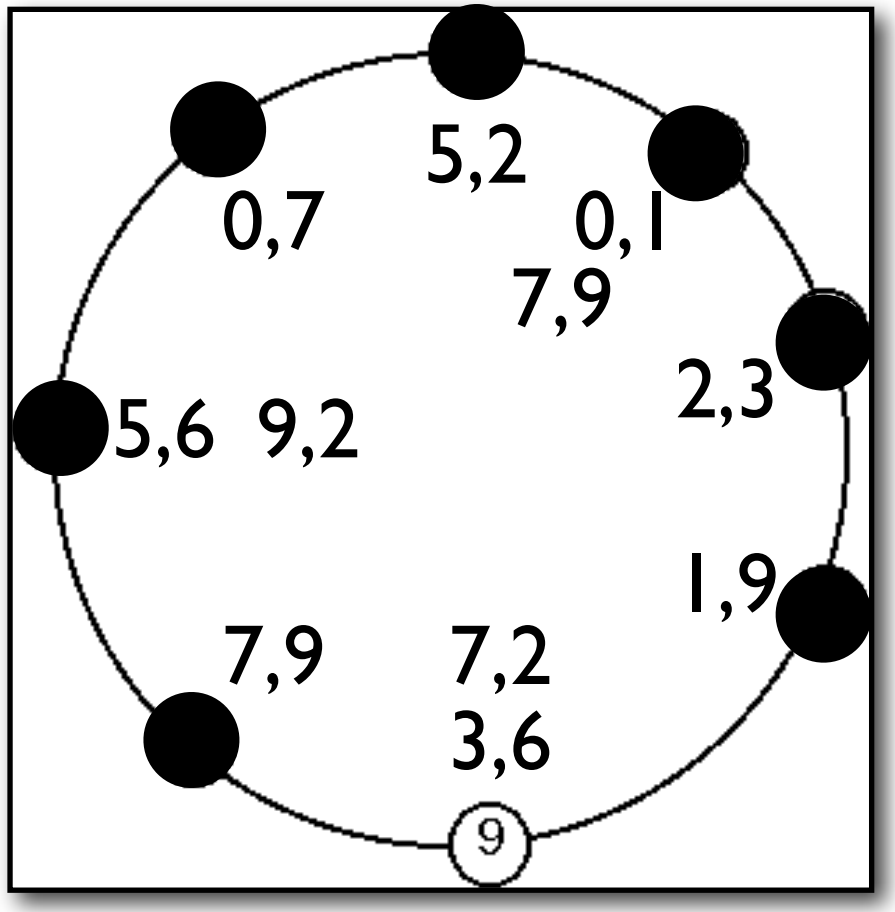
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



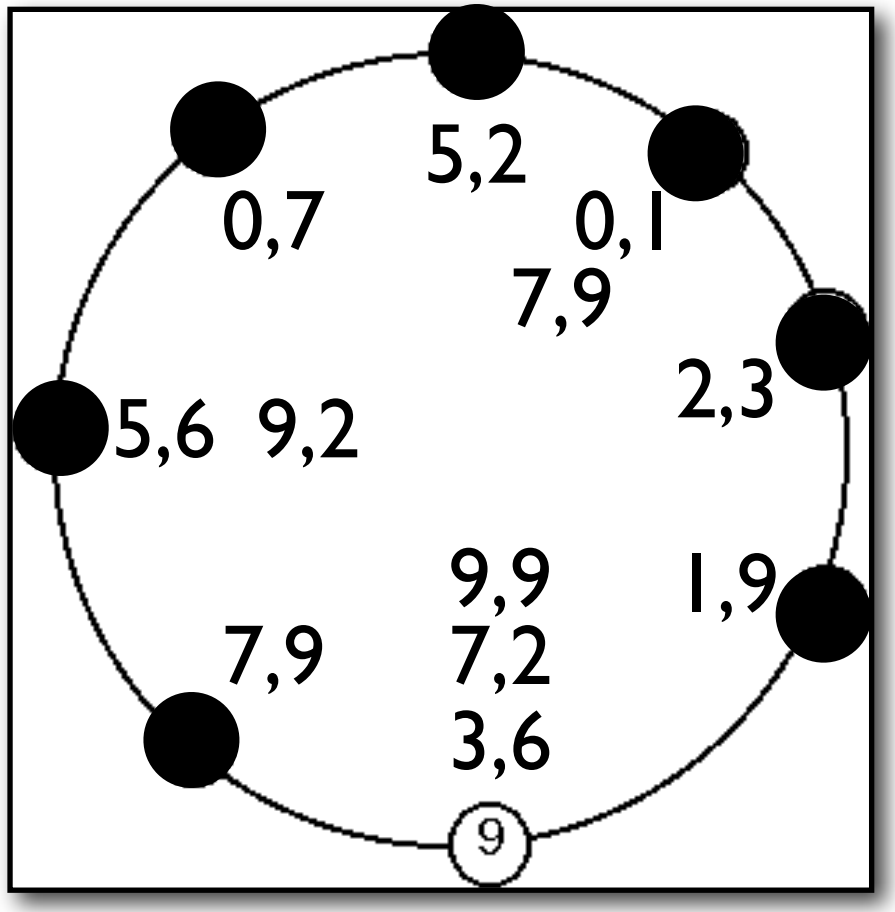
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



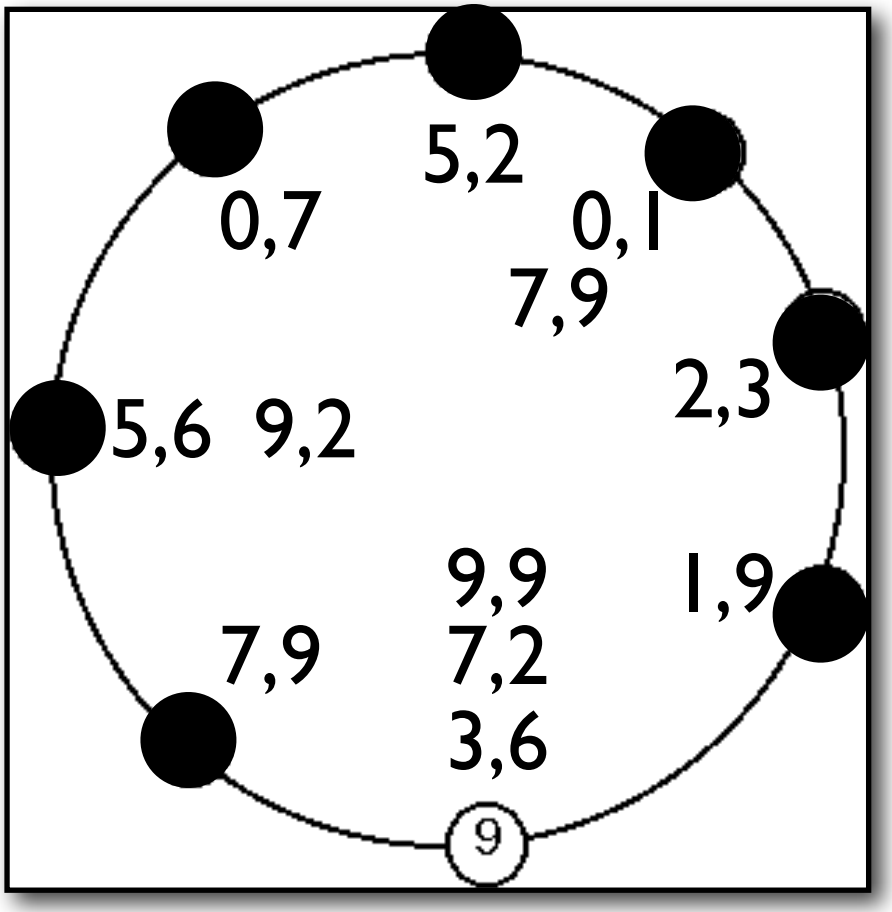
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



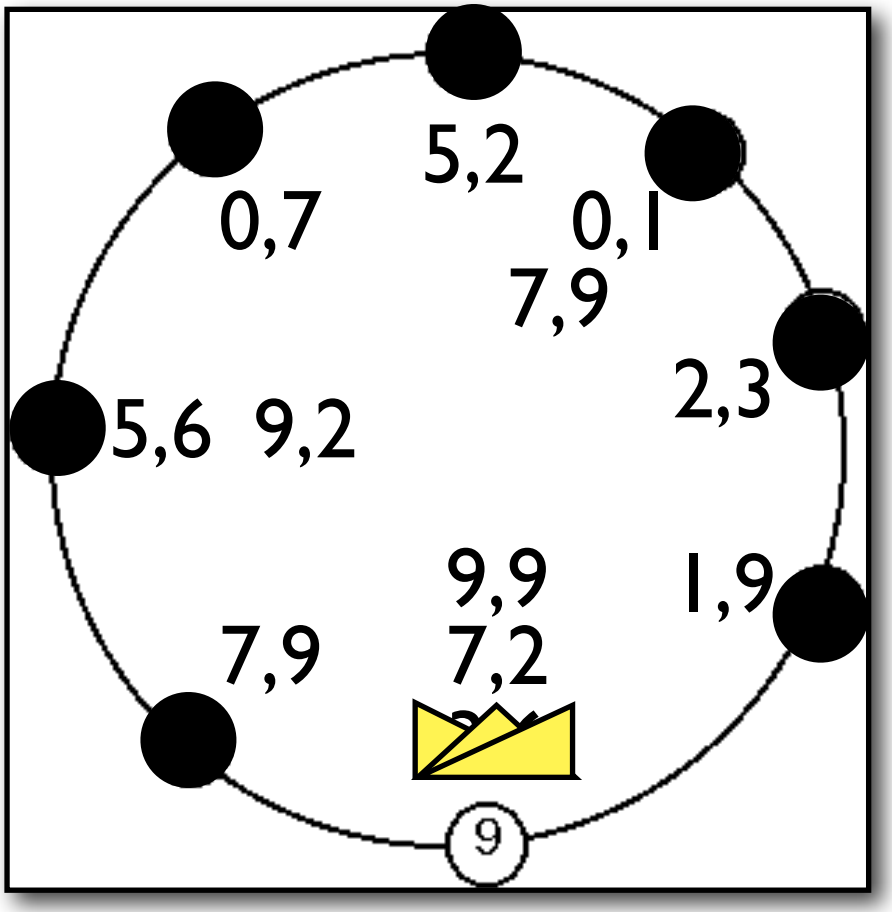
Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



Leader election in Netzen
im Ring - Beispiel

- sende beiden Nachbarn deine Information id
- empfange id_1 und id_2 von beiden Nachbarn
- falls $id_1 > id$ oder $id_2 > id$ dann werde passiv
- falls $id_1, id_2 < id$ dann alles nochmal (nächste Runde)
- falls $id_1 = id_2 = id$ dann erkläre dich zum Leader



Leader election in Netzen
Aufwand

Rechenaufwand

- Ring mit n Rechnern => mind. $^n/_2$ Rechner haben Nachbarn mit größerer id
- je Runde wird mind. die Hälfte der noch aktiven Rechner passiv
- max. log n Runden ⇒ nur noch ein Rechner aktiv

Kommunikationsaufwand

- je Runde sendet jeder Rechner 2 Nachrichten mit je log n Bits, insgesamt also $O(n (\log n)^2)$ Bits

Leader election in Netzen
anonyme Netze - kontrollierter Zufall

Problem

- es gibt keinen Leader-election Algorithmus für synchrone anonyme Rechnernetzringe
- Grund: nach jeder Runde sind alle Prozessoren im selben Zustand – Folge der Symmetrie

Lösung - Symmetry breaking

- kontrollierter Zufall löst die Symmetrie auf

Leader election in Netzen
Algorithmus mit kontrolliertem Zufall

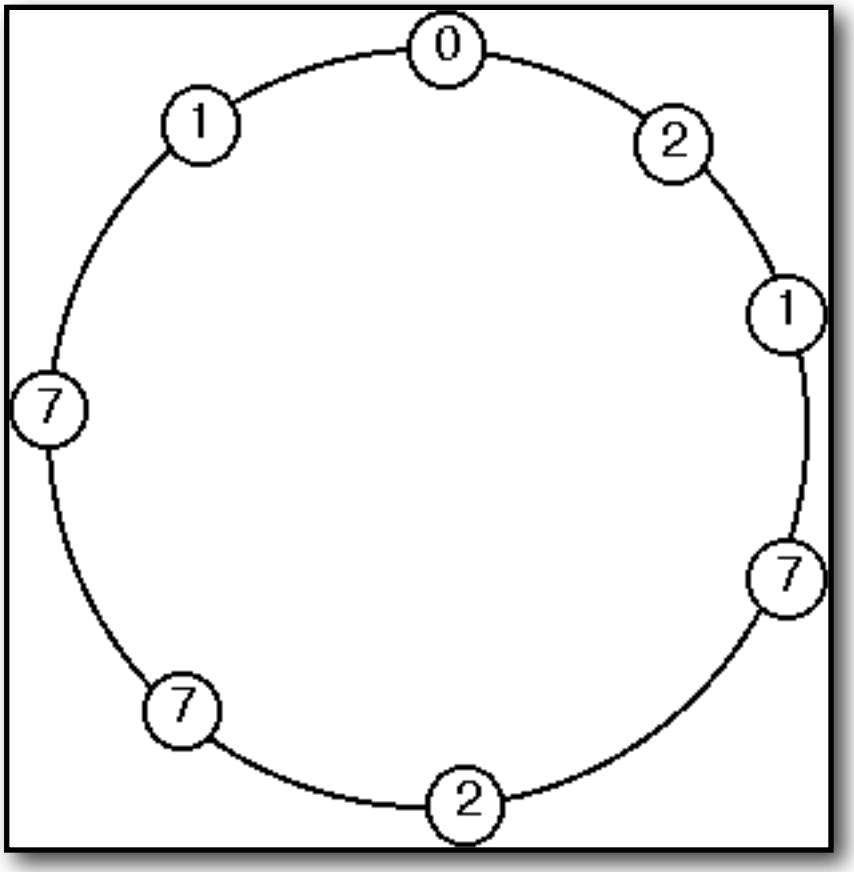
Programm für jeden Rechner:

- var s: list of {1,...,K}
- wiederhole solange bis eine id in s eindeutig ist
  - s:=∅
  - id sei **zufällige** Zahl aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn

Rechner mit größter eindeutiger id wird Leader.

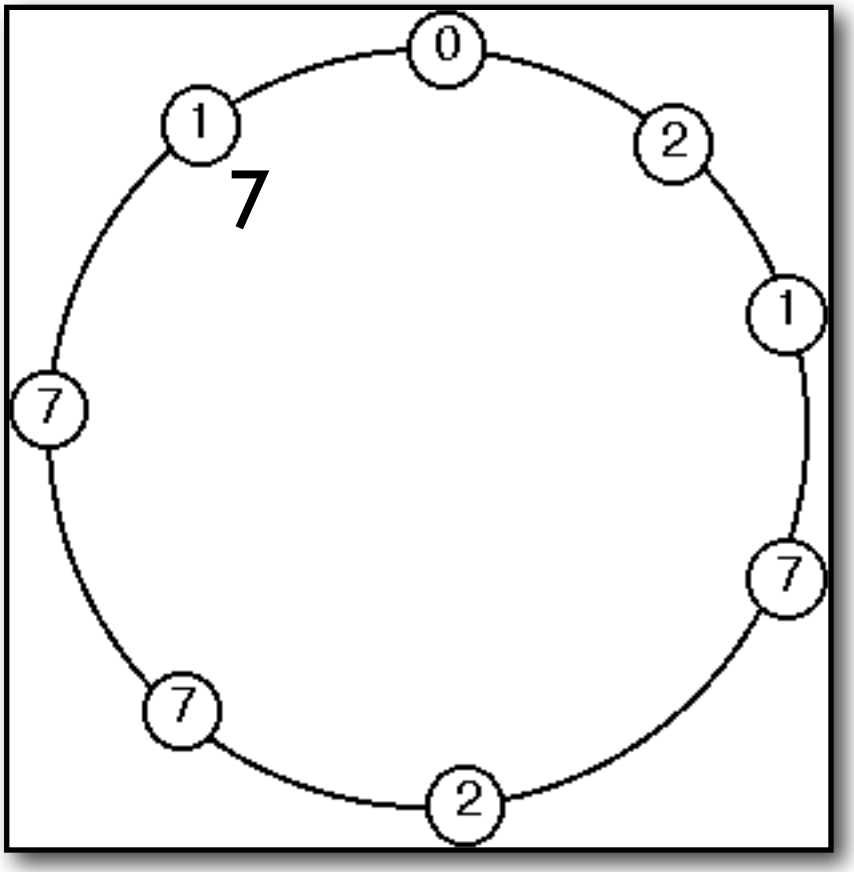
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



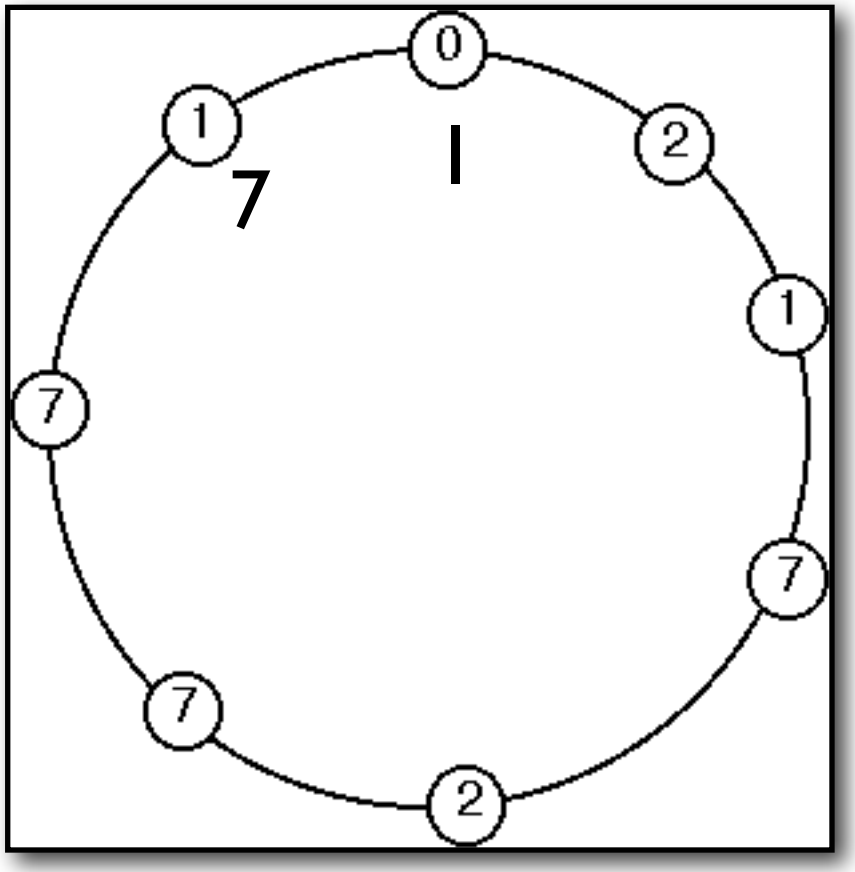
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



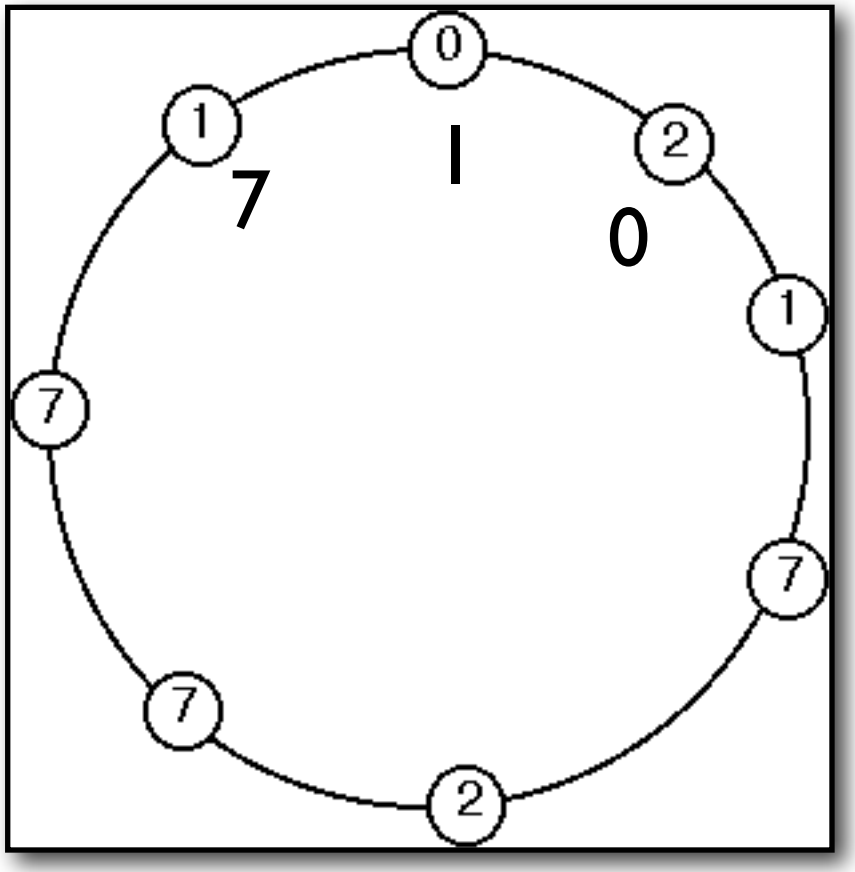
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



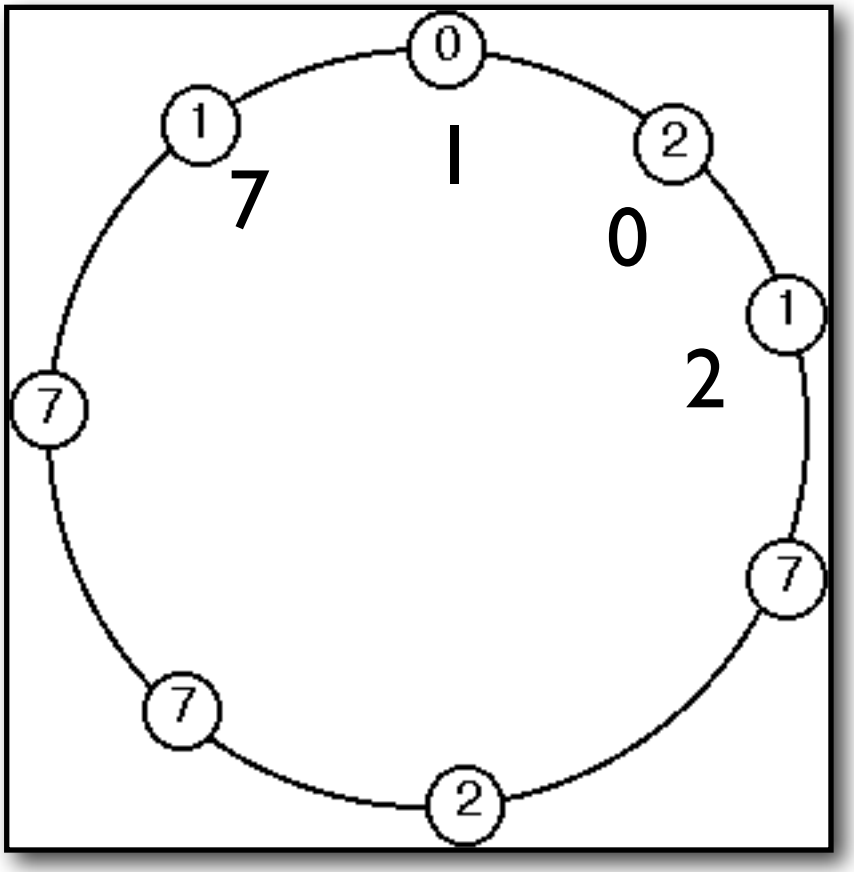
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



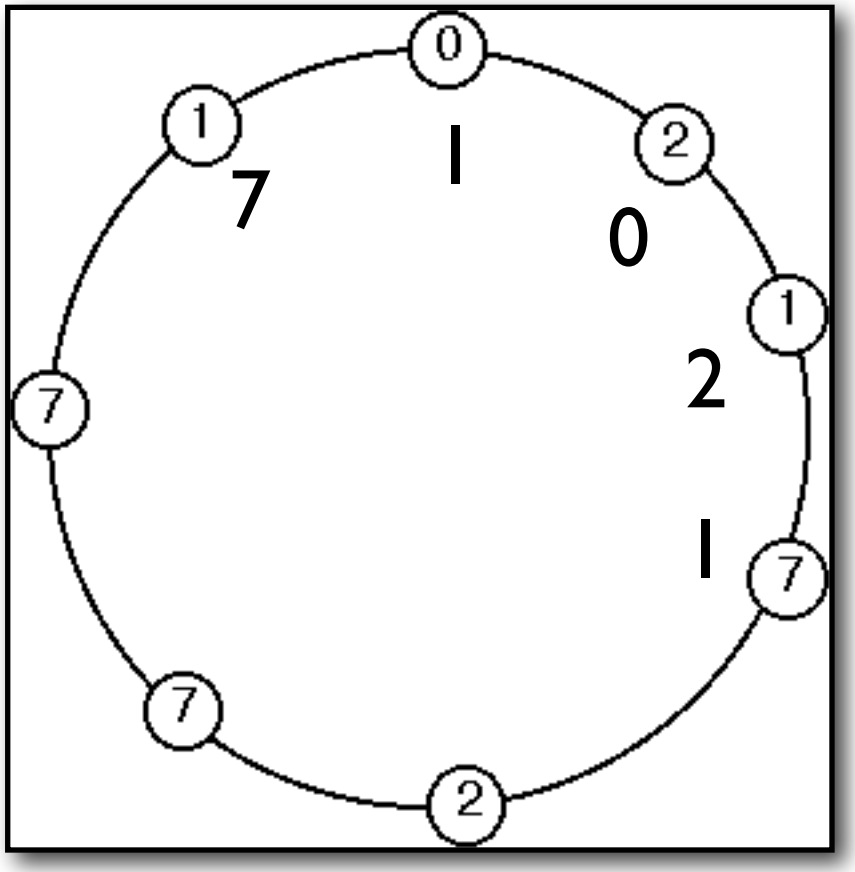
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



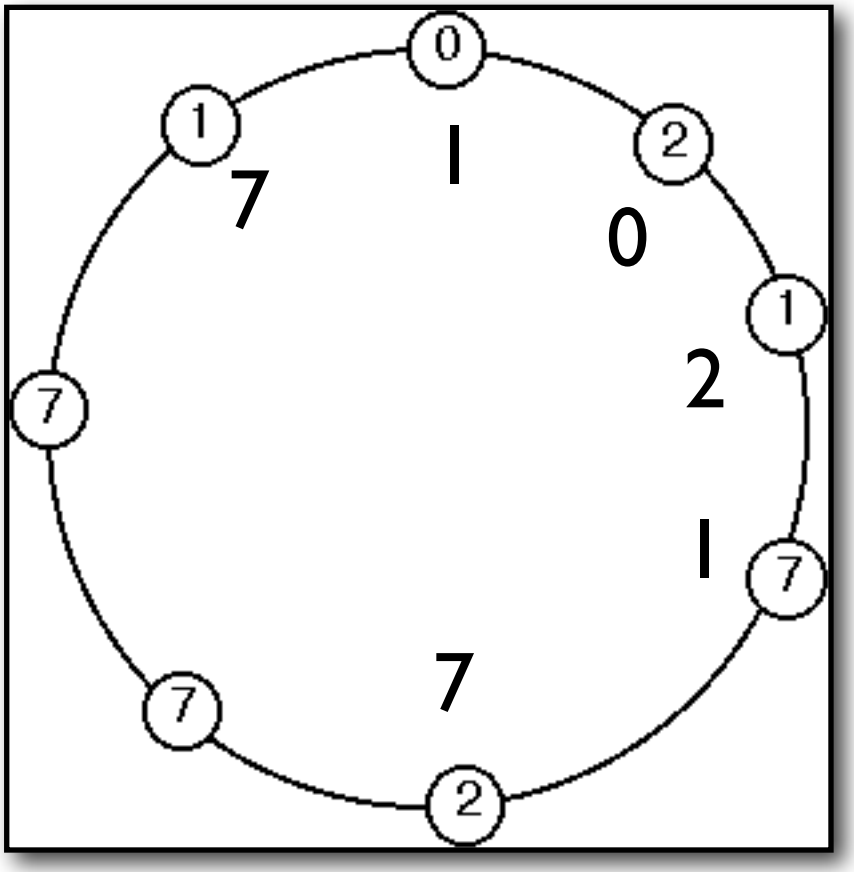
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



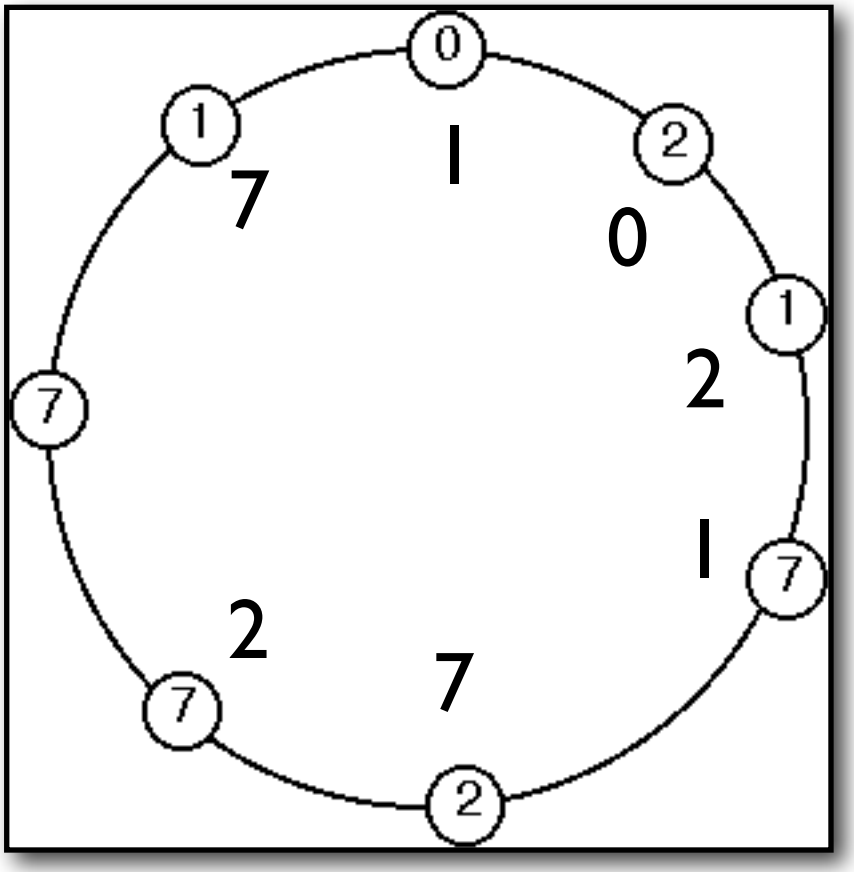
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



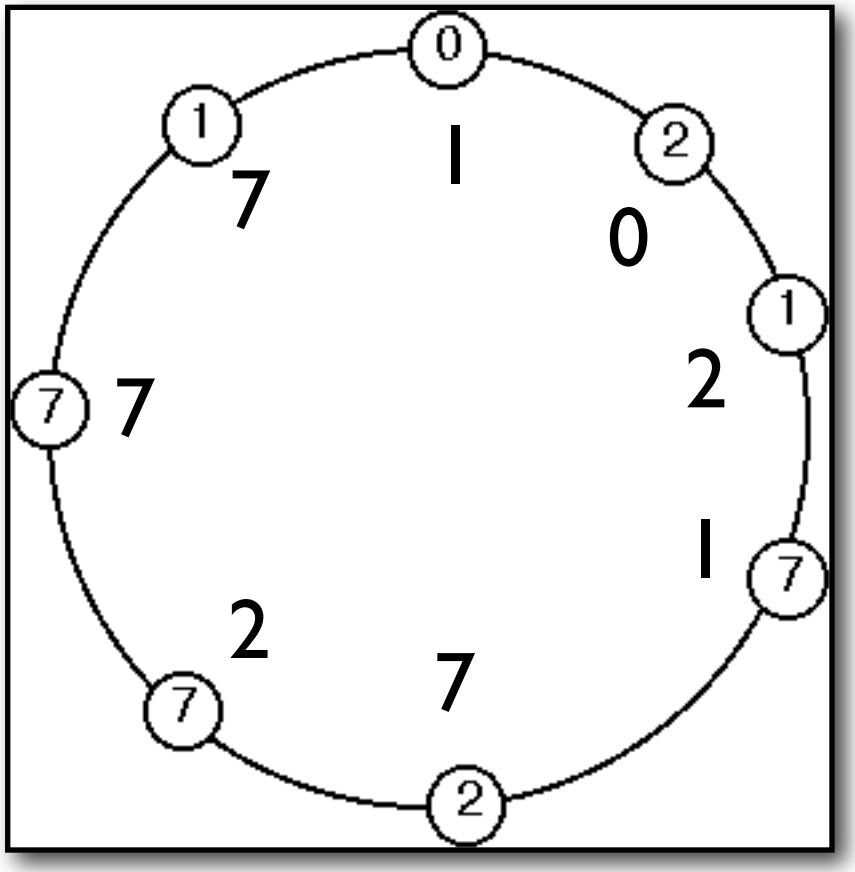
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



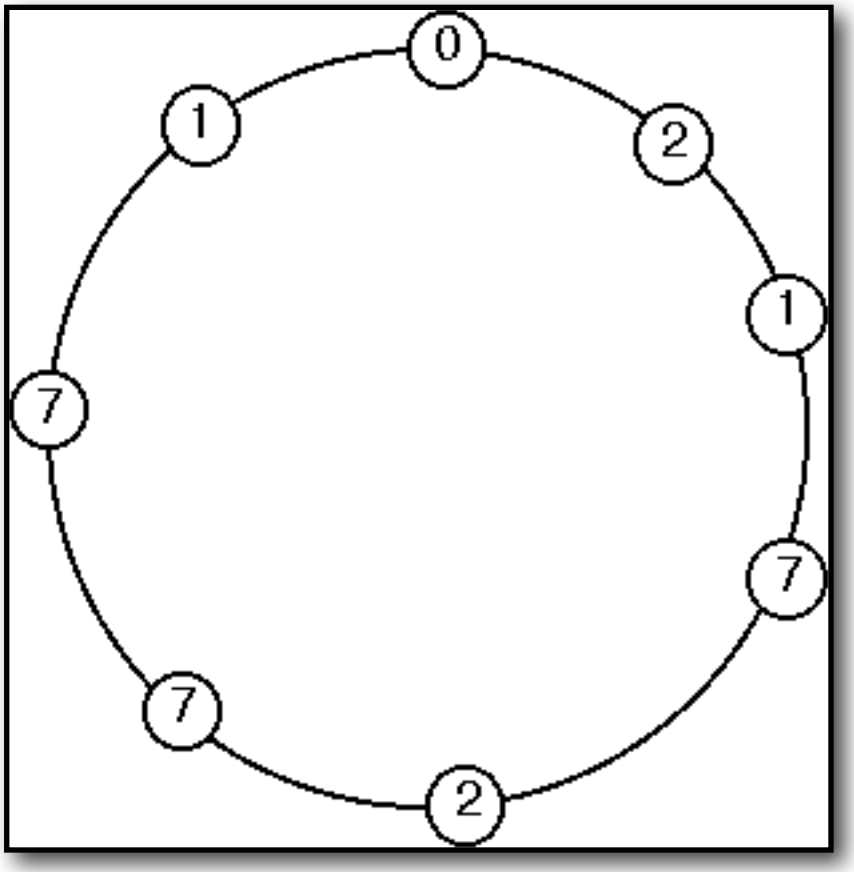
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



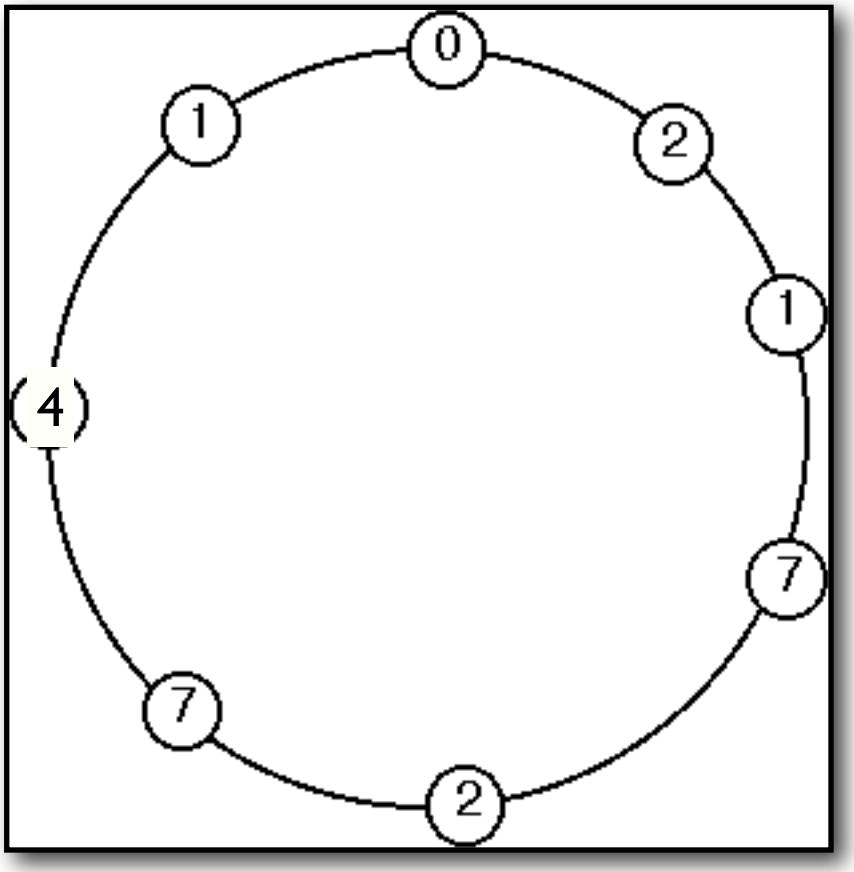
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



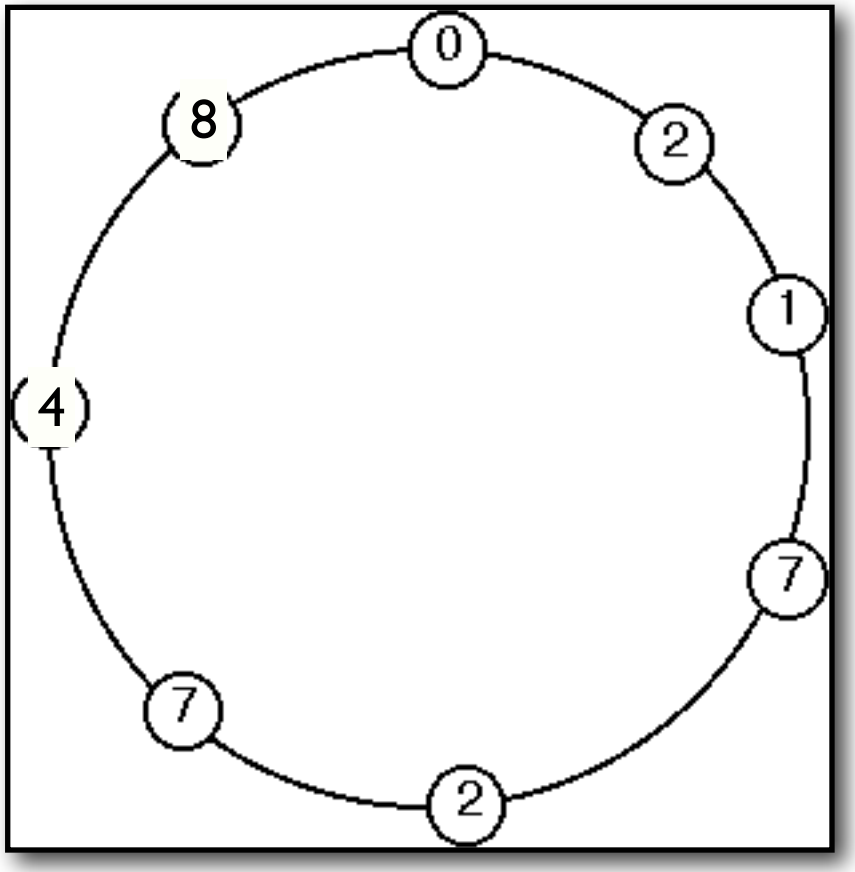
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



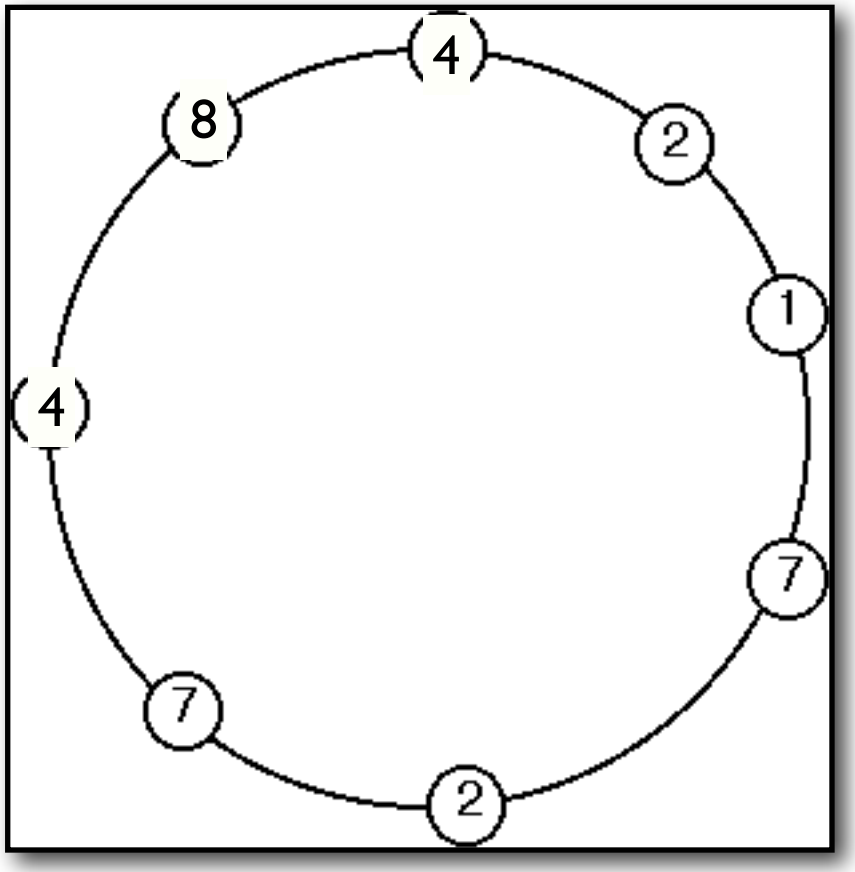
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



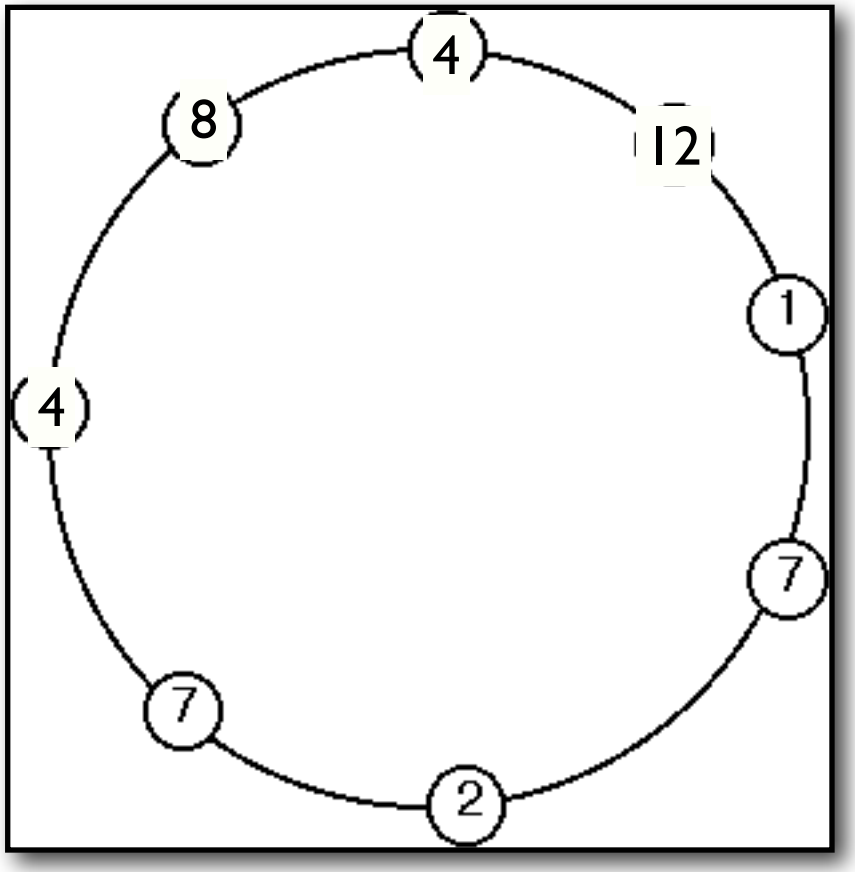
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



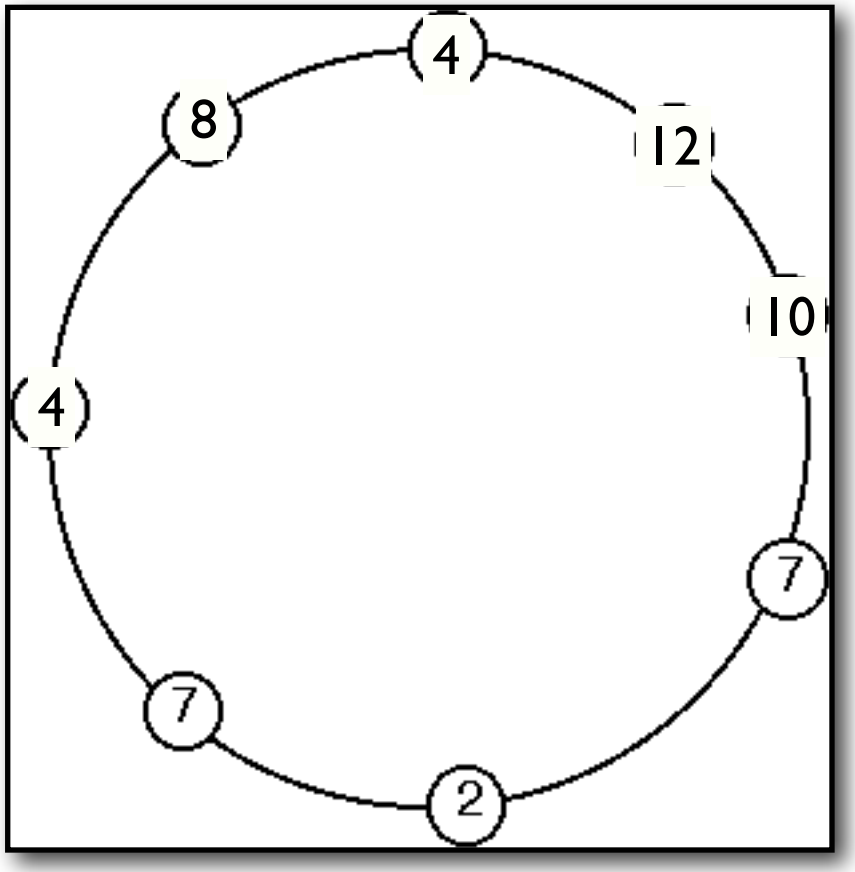
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



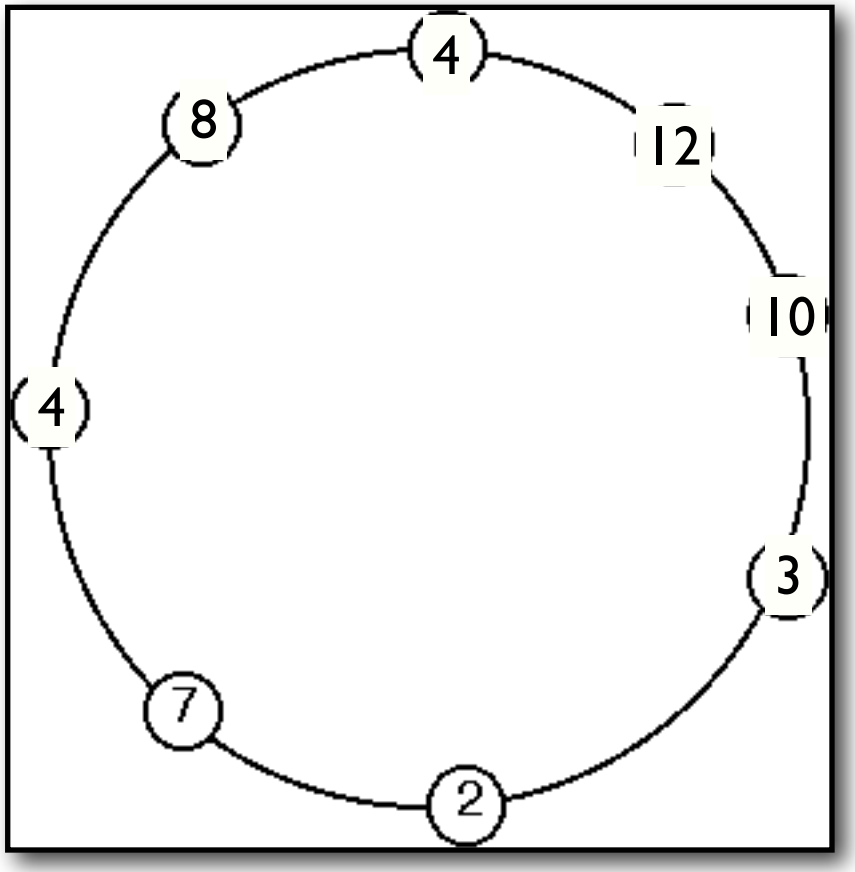
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



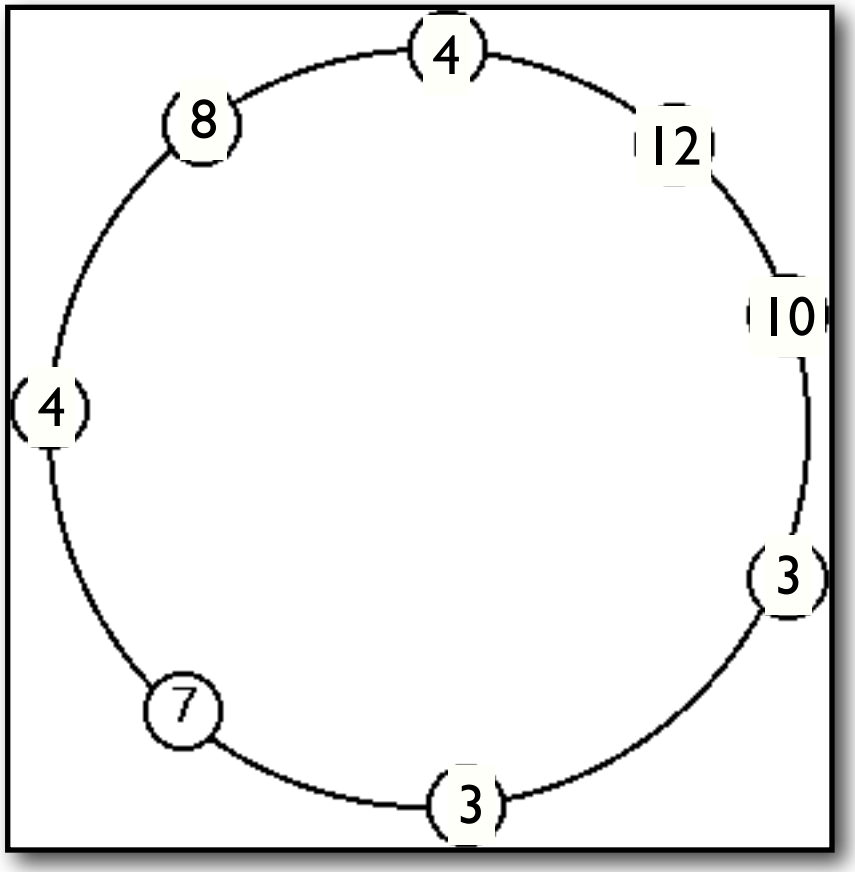
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



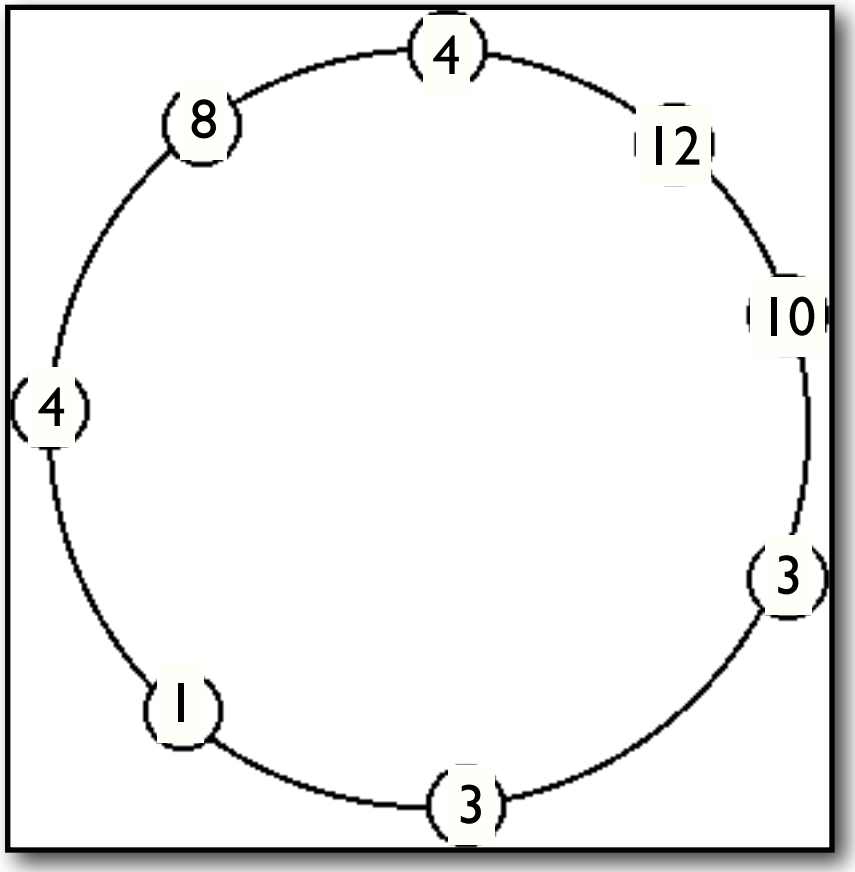
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



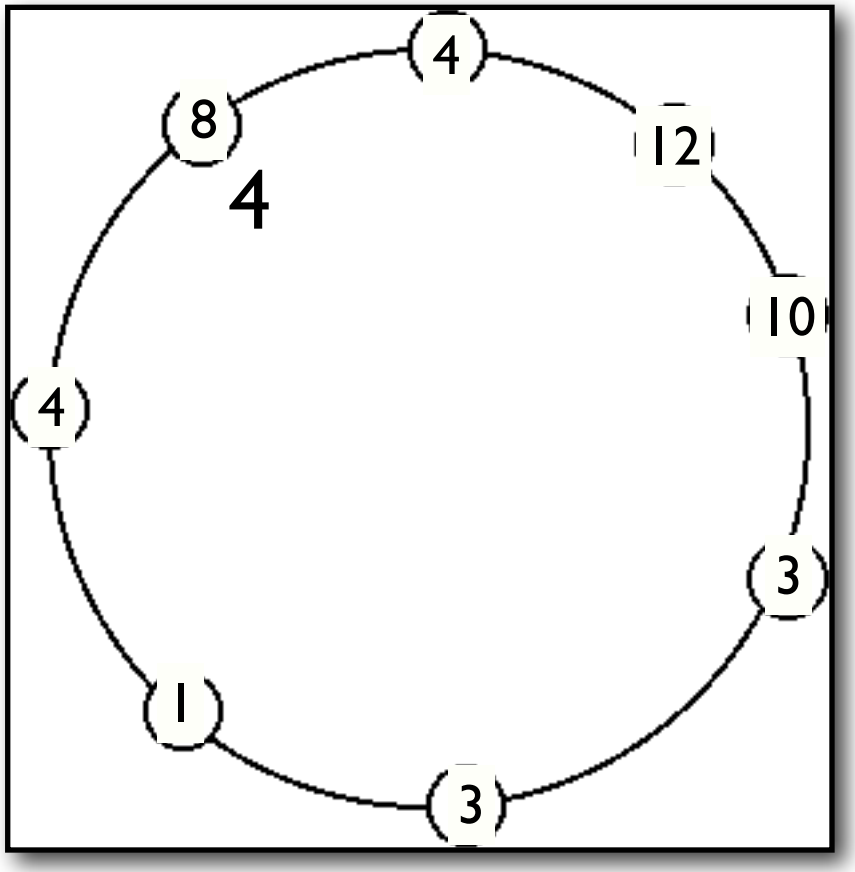
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



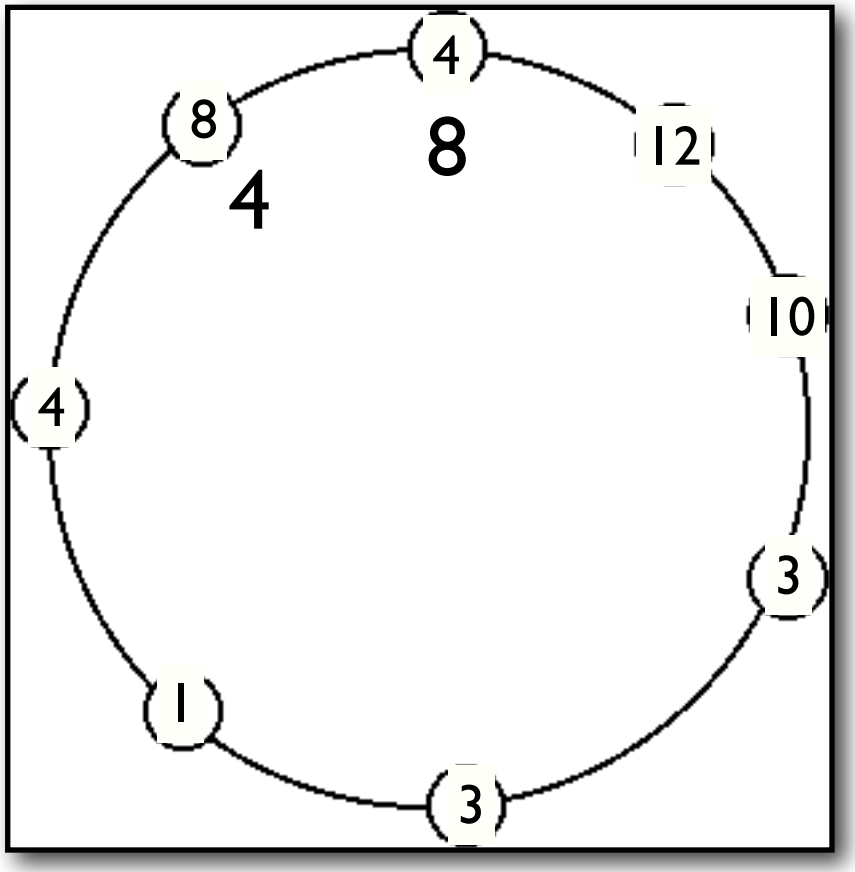
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



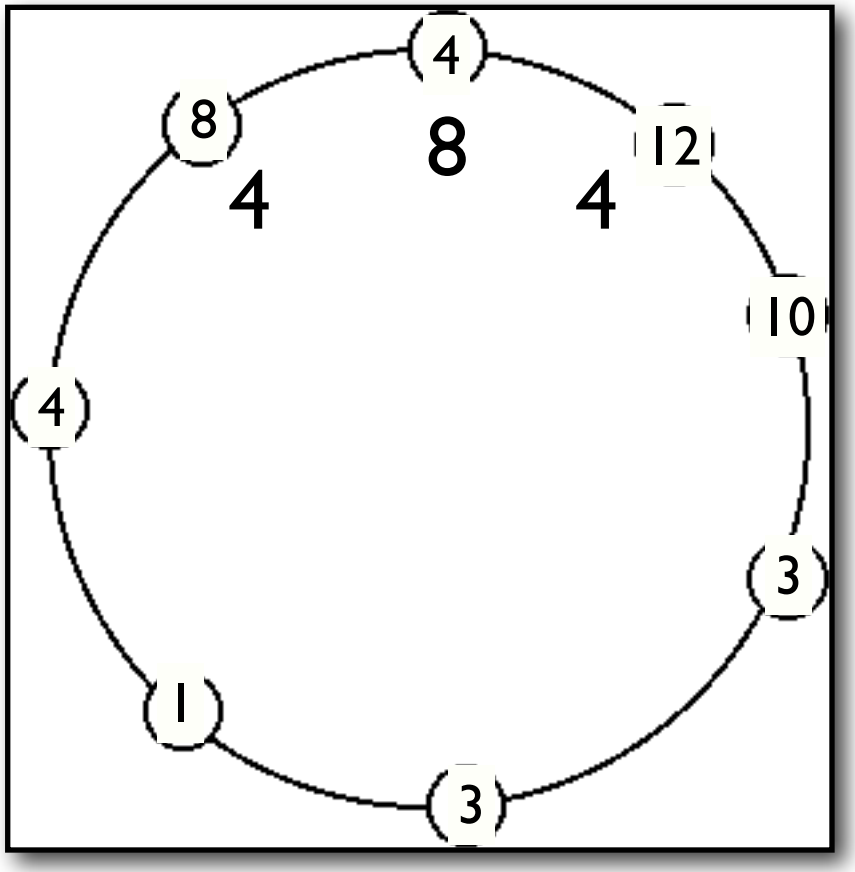
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



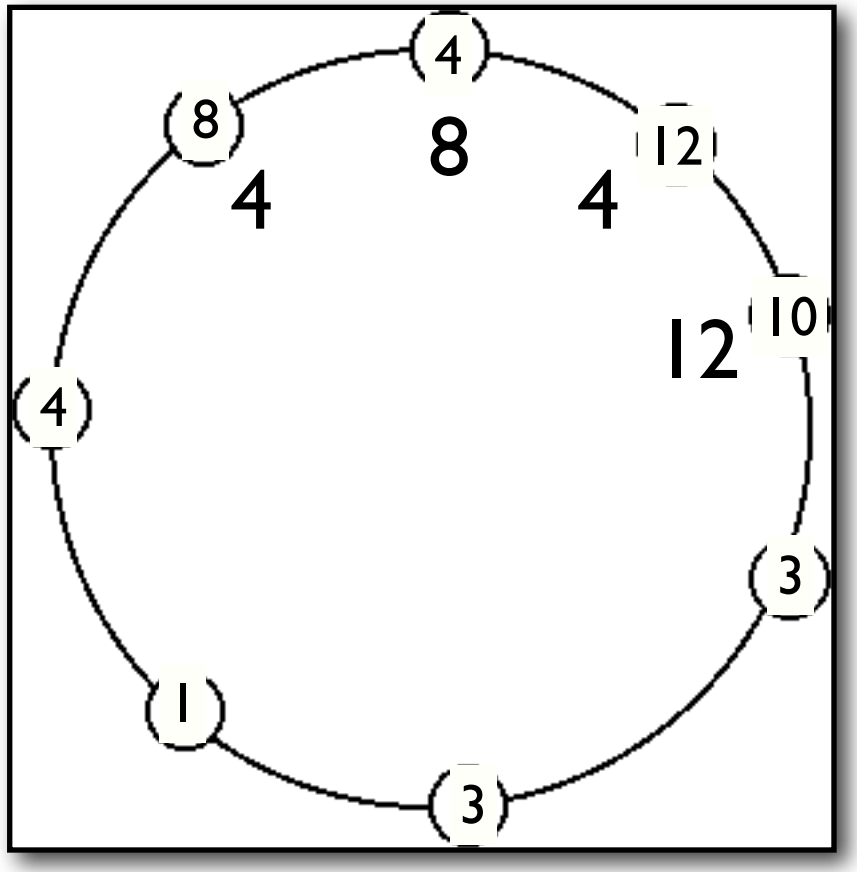
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



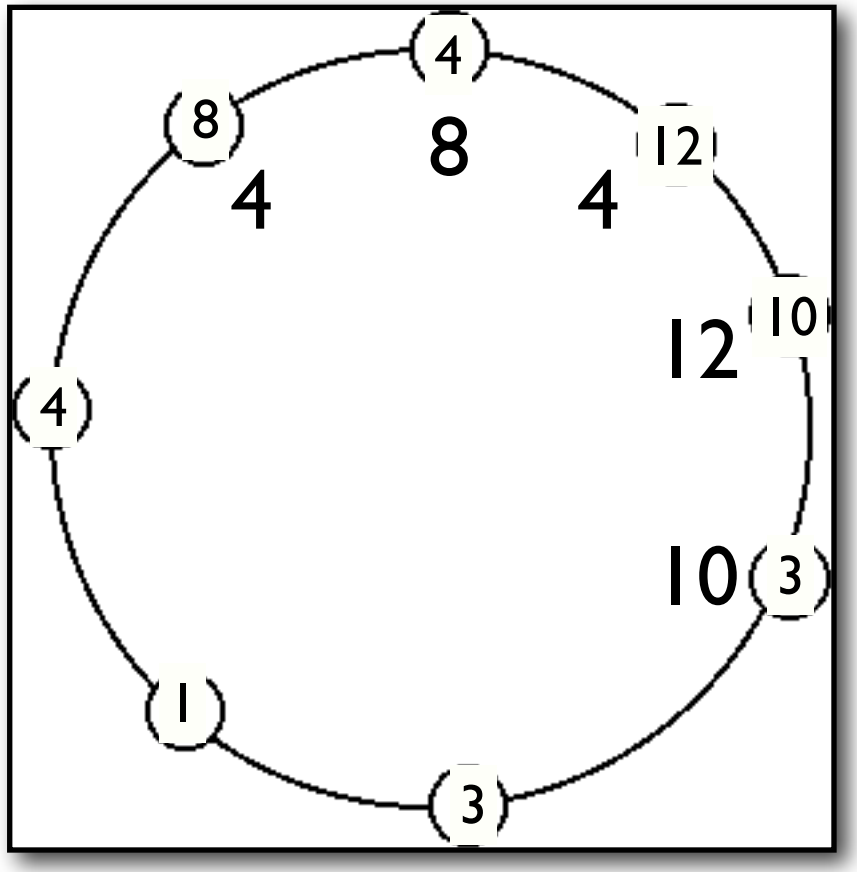
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



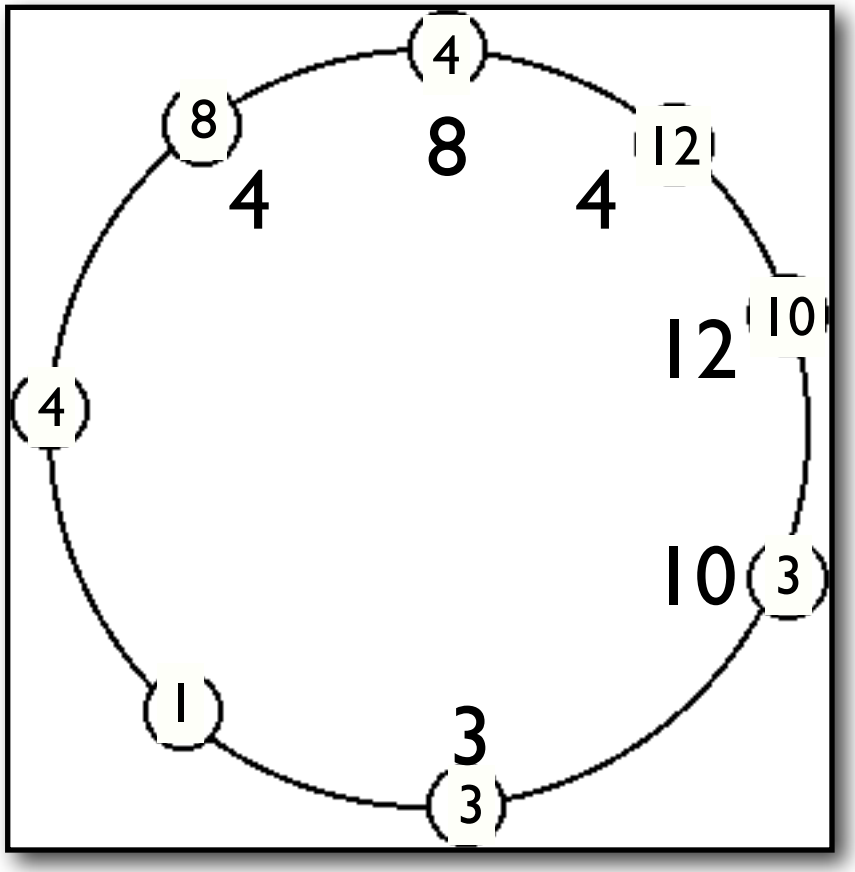
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



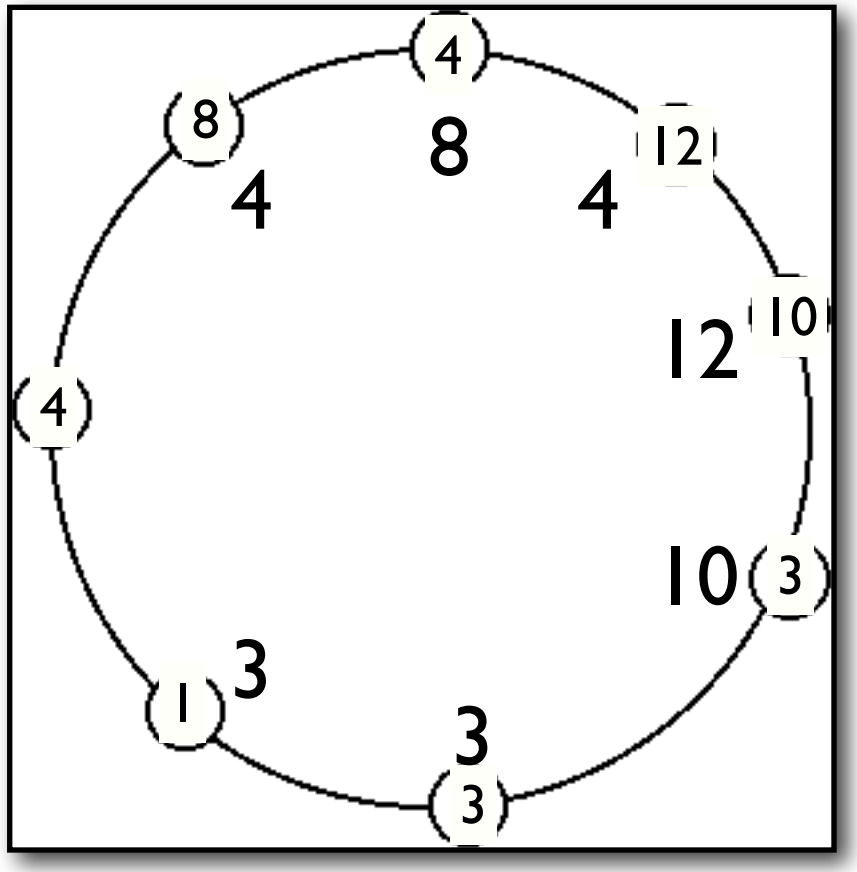
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



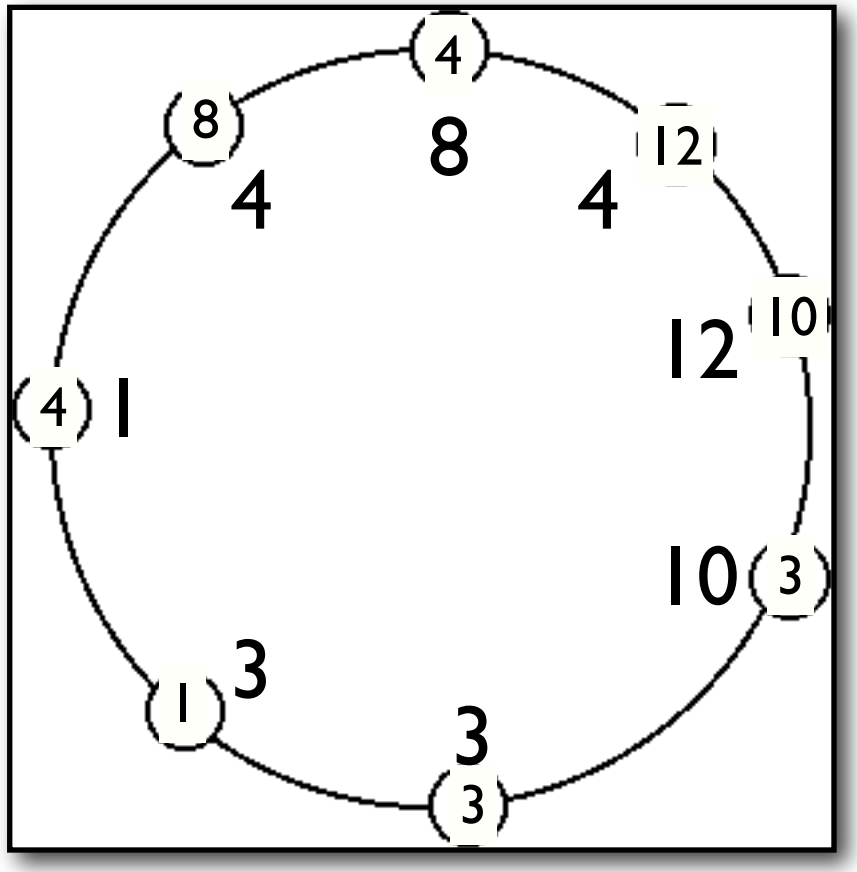
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



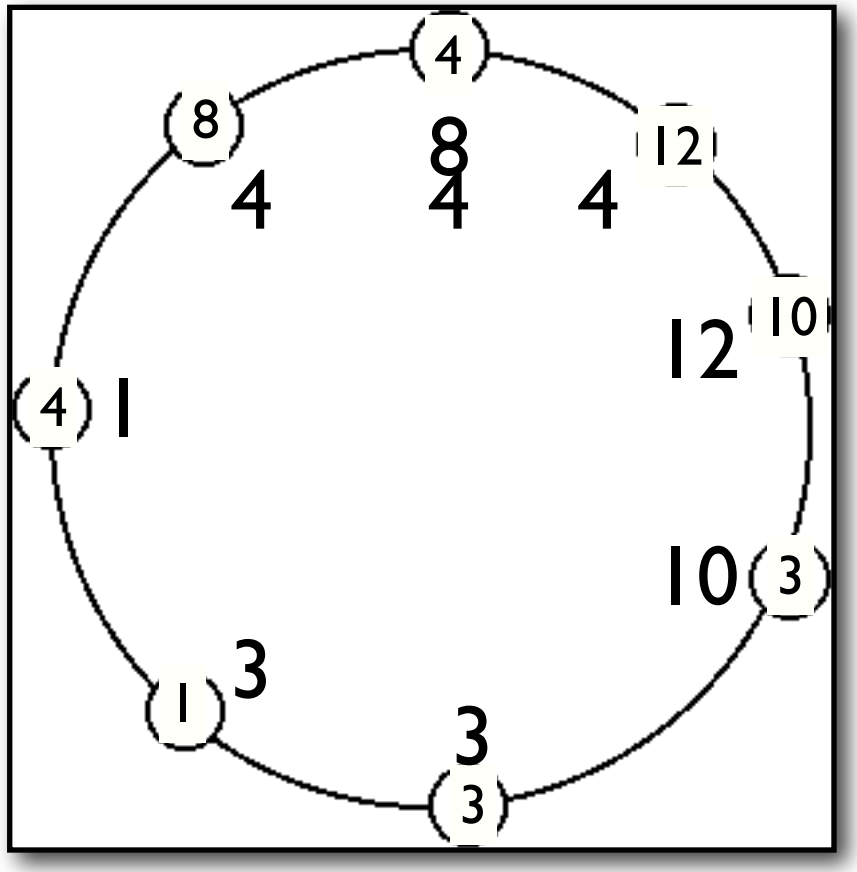
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



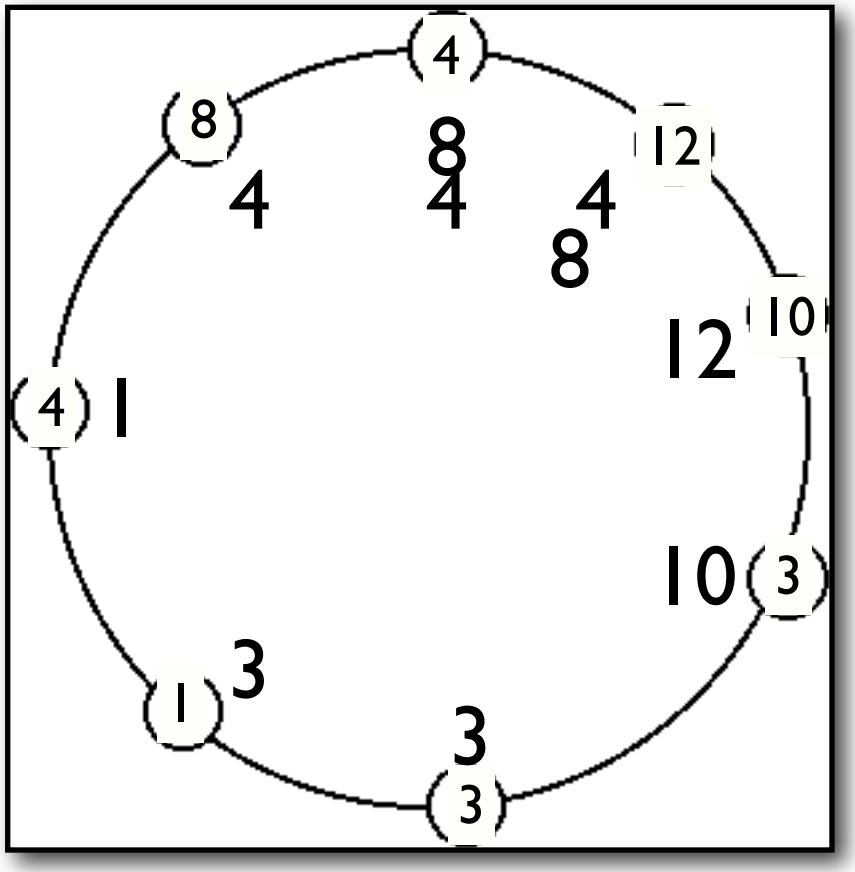
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



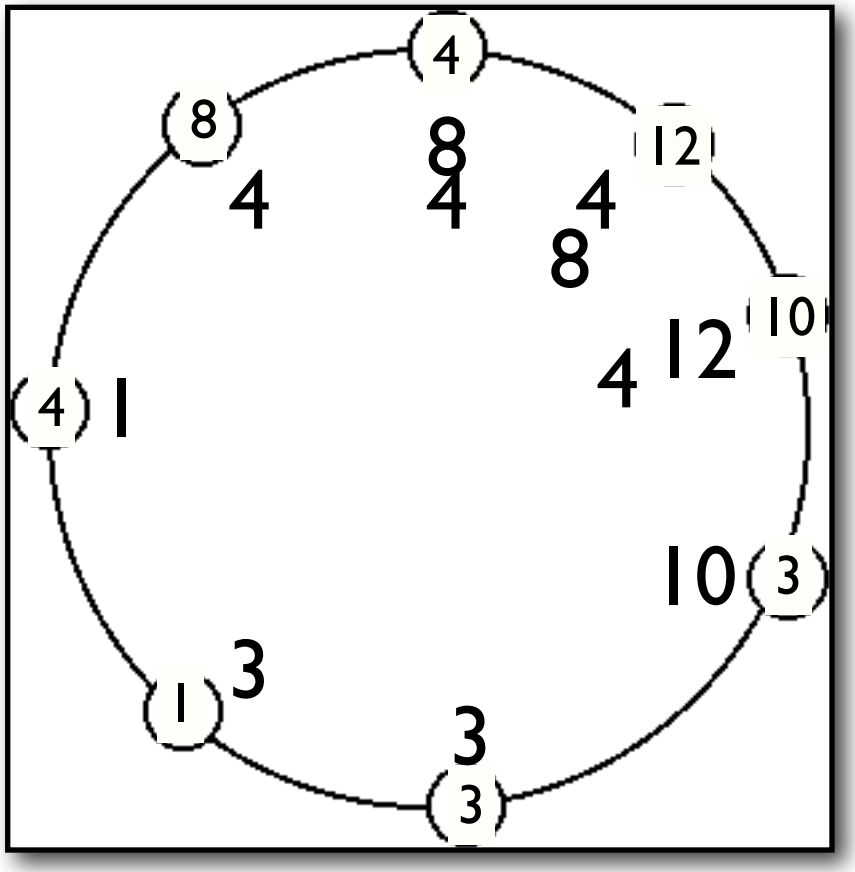
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



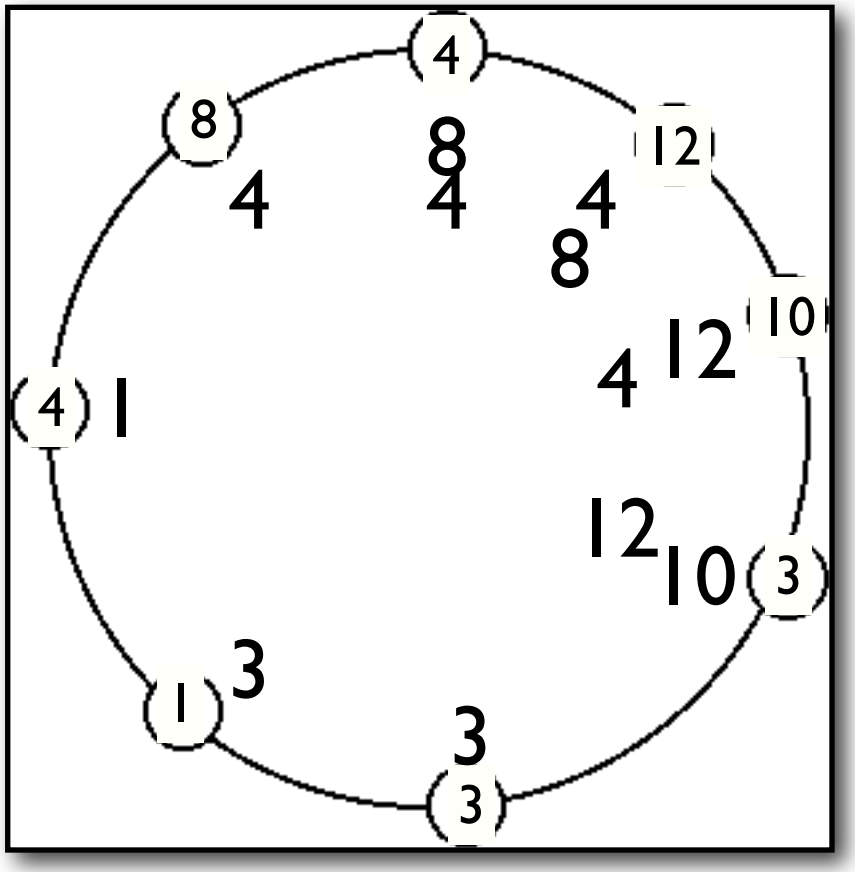
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



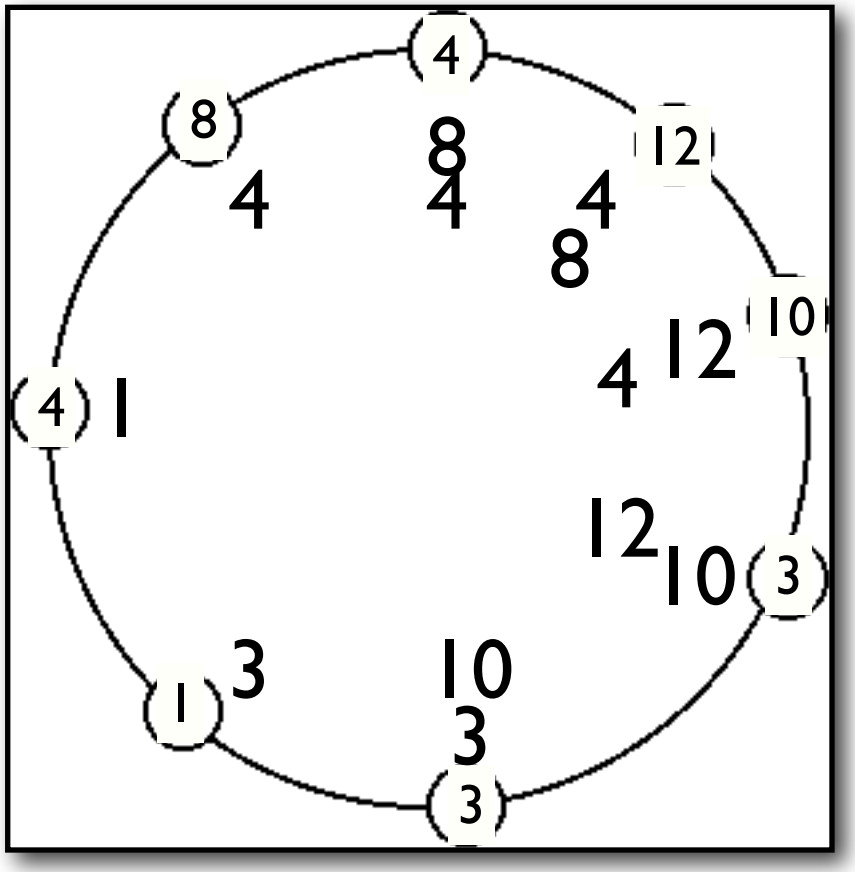
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



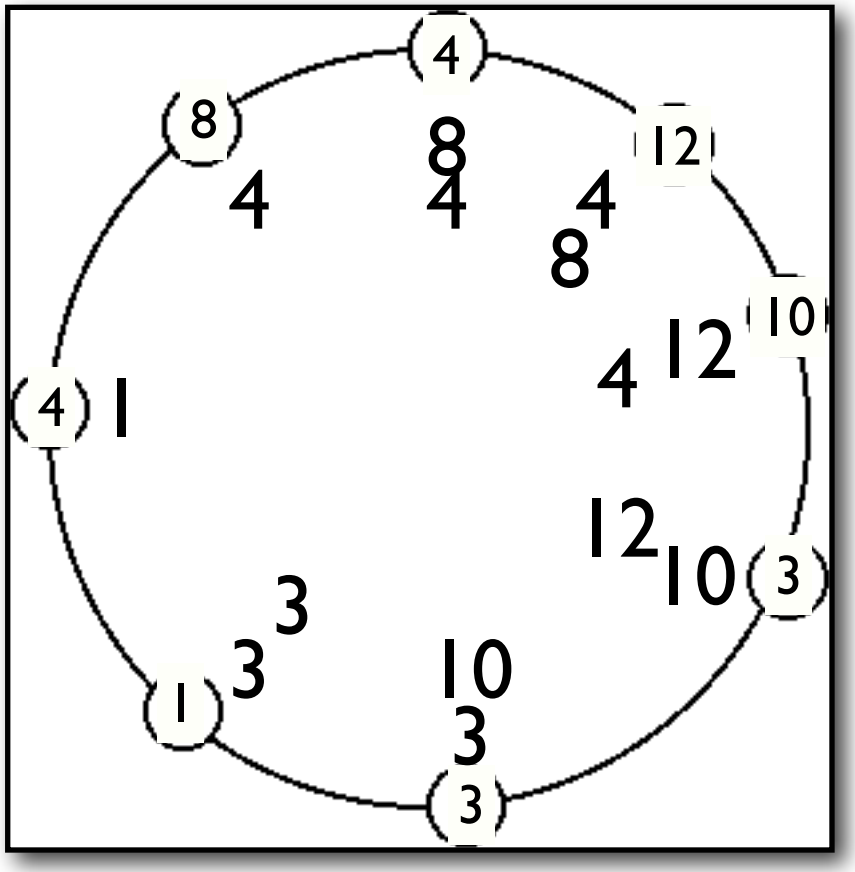
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



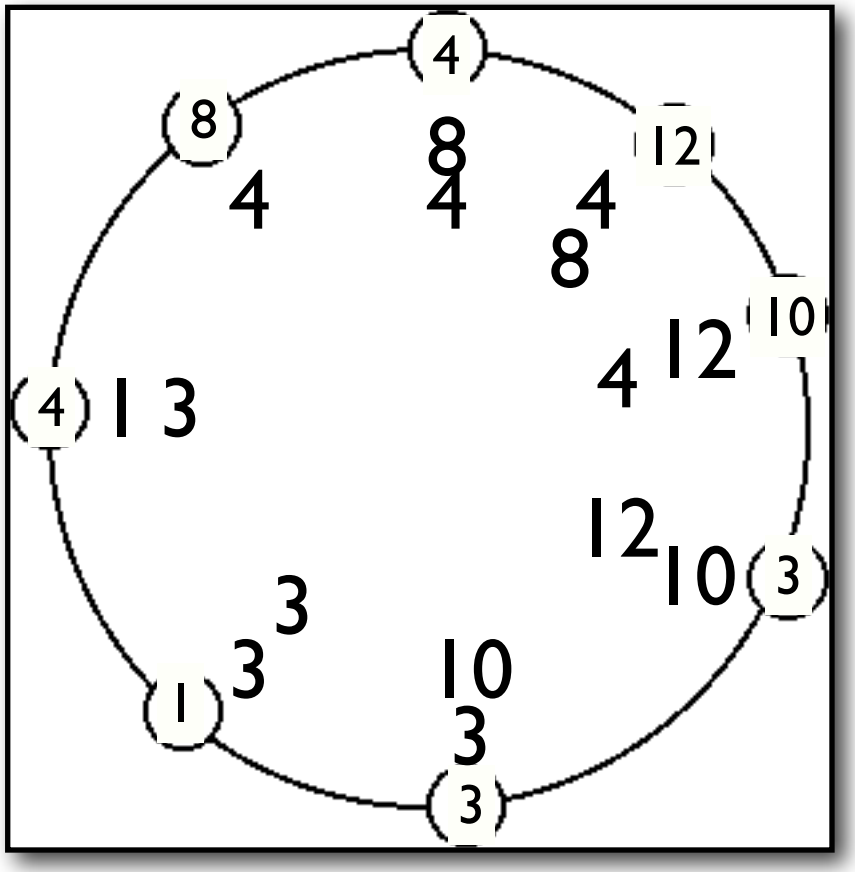
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



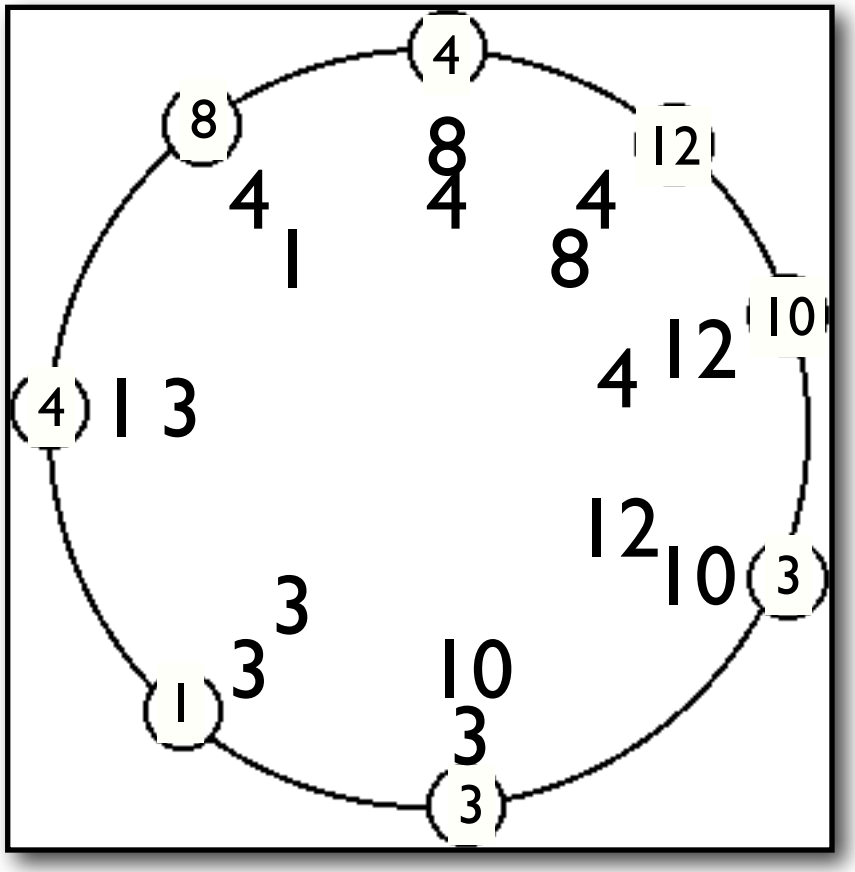
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



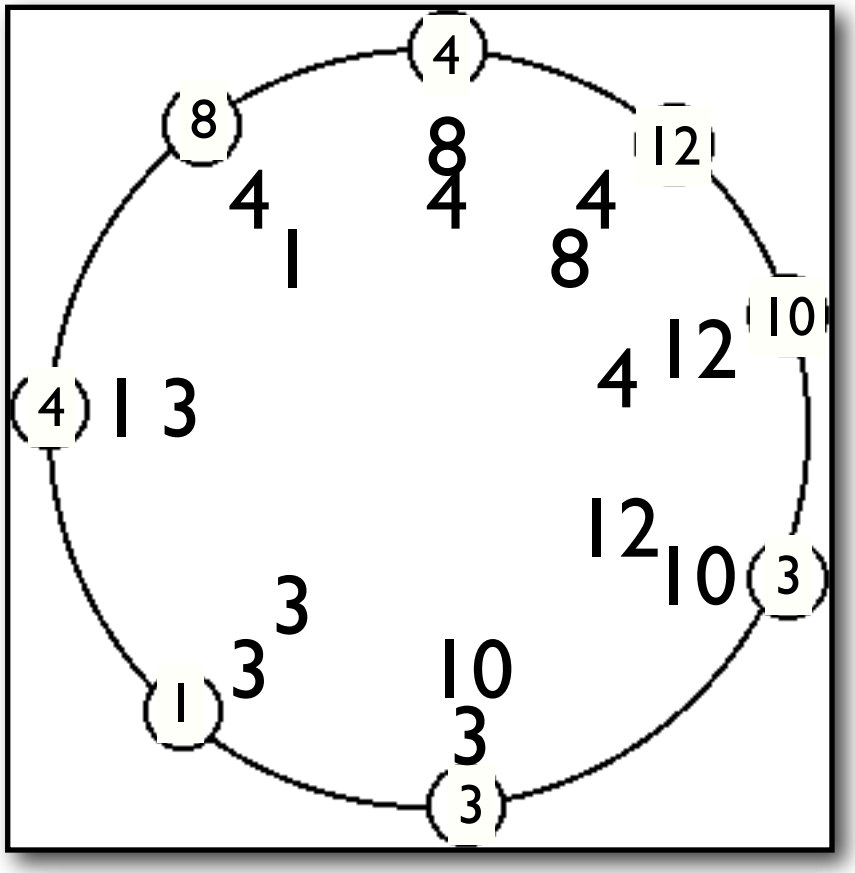
Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



Leader election in Netzen
im Ring - Beispiel

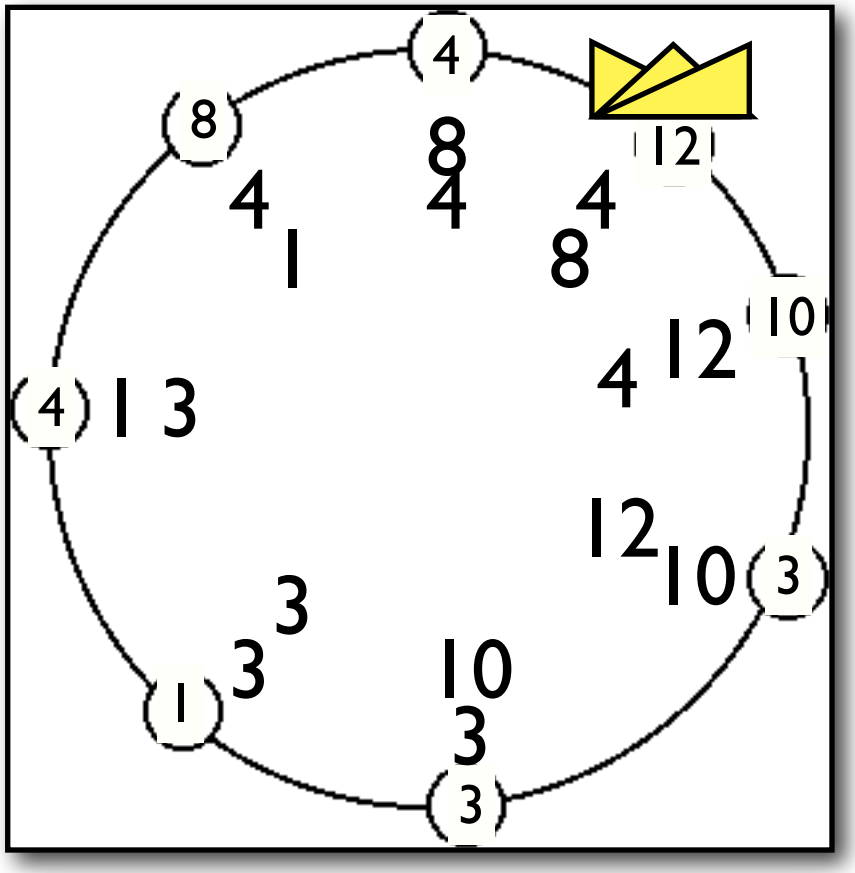
- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



Nach 8 Runden wird Rechner 12 Leader

Leader election in Netzen
im Ring - Beispiel

- var s: list of {1,...,K}
- wiederhole bis eine id in s eindeutig
  - s:=∅
  - id **zufällig** aus {1,...K}
  - wiederhole n-mal
    - füge id zu s hinzu
    - sende id an linken Nachbarn
    - empfange id vom rechten Nachbarn



Nach 8 Runden wird Rechner 12 Leader

Leader election in Netzen
mit Zufall - Aufwand

Rechenaufwand

"wiederhole bis eine id in s eindeutig"

- entscheidet über Laufzeit des Algorithmus
- Wahrscheinlichkeit geht gegen 1.

Kommunikationsaufwand

- wie oben, abhängig von der Zahl der Runden