Leader Election II

Algorithmen für verteilte Systeme

Sebastian Forster

Universität Salzburg

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Follower*
- **Gesucht:** Algorithmus, der für jeden Knoten entscheidet, ob er der Leader ist oder ein Follower
- **Motivation:** Leader kann Koordinationsaufgaben übernehmen

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Follower*
- **Gesucht:** Algorithmus, der für jeden Knoten entscheidet, ob er der Leader ist oder ein Follower
- **Motivation:** Leader kann Koordinationsaufgaben übernehmen

Symmetry Breaking:

- A priori eignet sich jeder Knoten als Leader
- Lösung ist nicht eindeutig, es gibt n verschiedene Lösungen
- Schwierigkeit beim Finden einer Lösung ist die konsistente Entscheidung für eine der Lösungen

Was bisher geschah…

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen

Was bisher geschah…

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Heute: Randomisierter Algorithmus

Was bisher geschah…

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Heute: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten

Was bisher geschah…

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Heute: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
3. Wartezeit Algorithmus:
   - Deterministischer synchroner Algorithmus für non-uniforme Ringe
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten

Was bisher geschah…

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Heute: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
3. Wartezeit Algorithmus:
   - Deterministischer synchroner Algorithmus für non-uniforme Ringe
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten
   - Frage: Effizienz in beiden Metriken möglich?

Was bisher geschah…

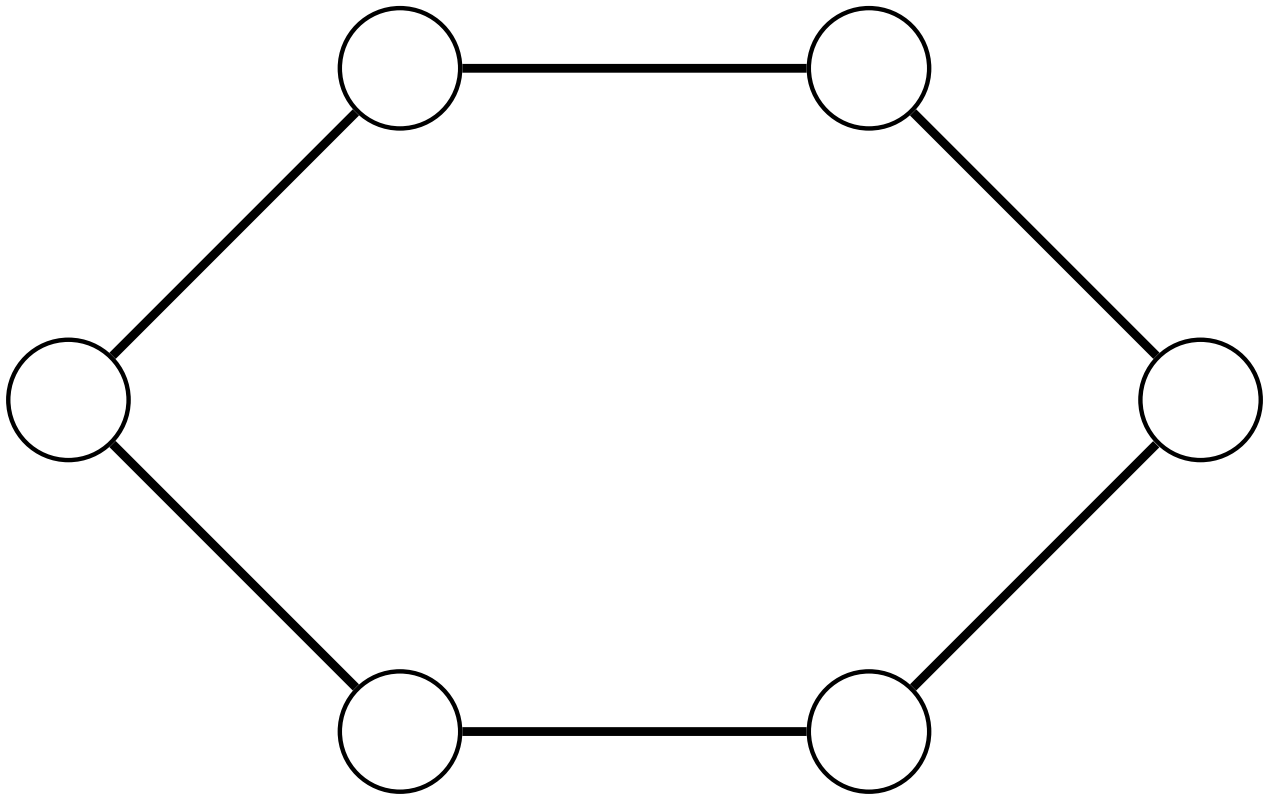
Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Heute: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
3. Wartezeit Algorithmus:
   - Deterministischer synchroner Algorithmus für non-uniforme Ringe
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten
   - Frage: Effizienz in beiden Metriken möglich?

   Heute: $O(n)$ Runden, $O(n \log n)$ Nachrichten

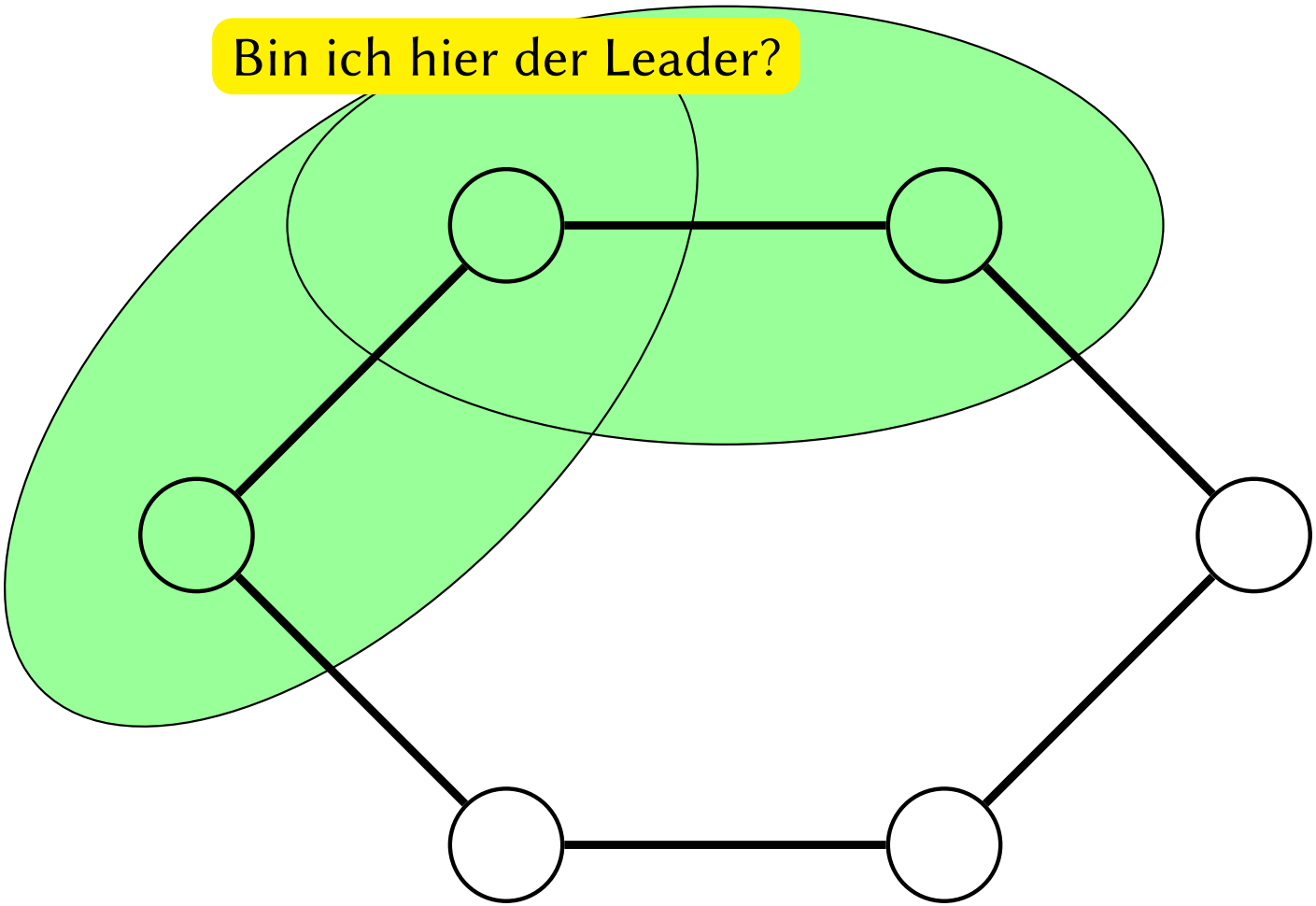
Idee: Radius Growth

Annahmen: synchron, identifizierbar, non-uniform



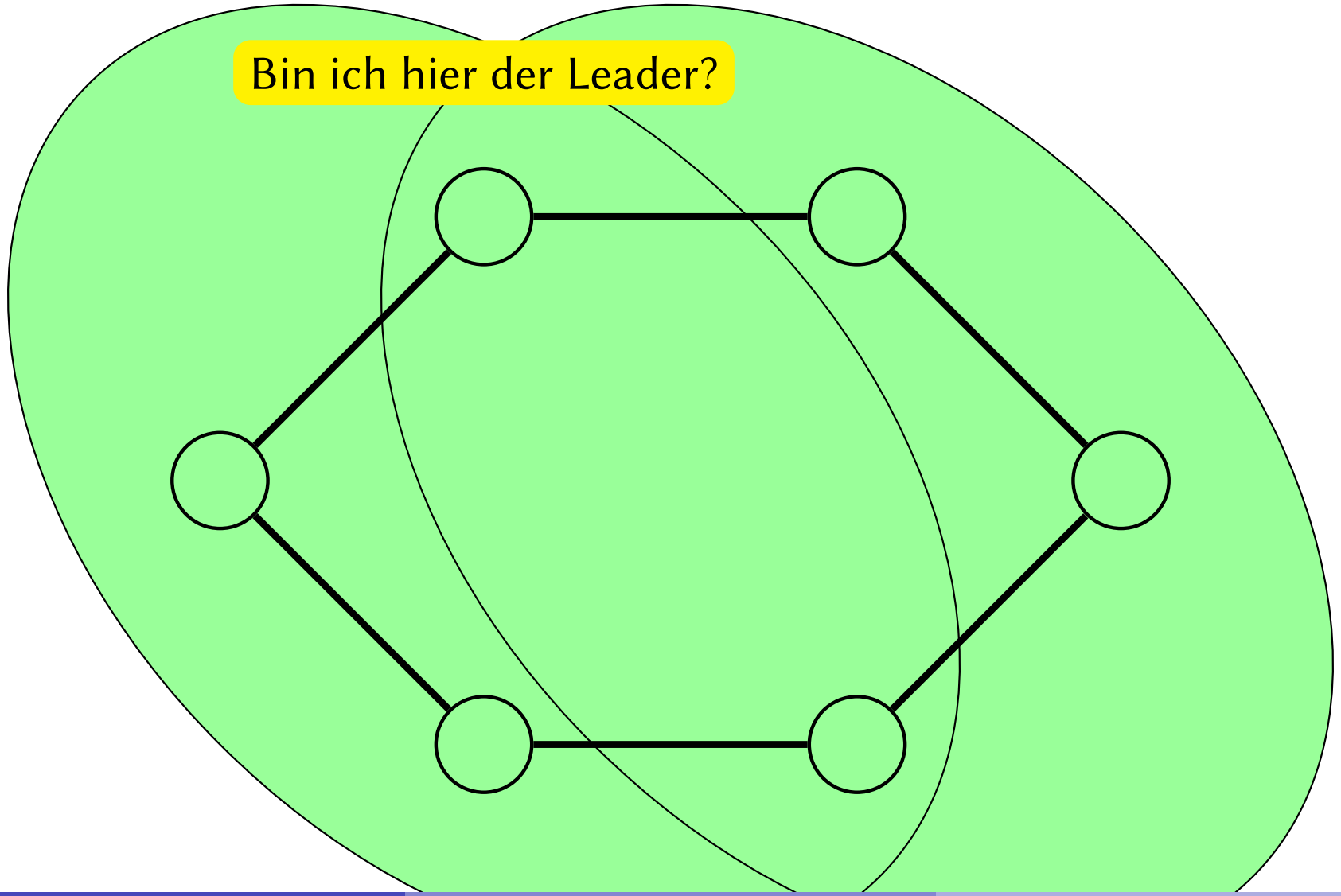
Idee: Radius Growth

Annahmen: synchron, identifizierbar, non-uniform



Idee: Radius Growth

Annahmen: synchron, identifizierbar, non-uniform



Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i-te Phase dauert $2^{i-1} + 1$ Runden

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i-te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v:

Erste Runde jeder Phase:

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i-te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v:

Erste Runde jeder Phase:

1 **if** v *noch kein Follower* **then**

2 $\quad$ v sendet $\mathrm{ID}(v)$ an Nachbarn im und gegen den Uhrzeigersinn

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i-te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v:

Erste Runde jeder Phase:

1 **if** v *noch kein Follower* **then**
2 $\quad$ v sendet $\mathrm{ID}(v)$ an Nachbarn im und gegen den Uhrzeigersinn

Jede andere Runde:

1 **if** v *empfängt Nachricht* M *von Nachbar gegen (im) den Uhrzeigersinn* **then**

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i-te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v:

Erste Runde jeder Phase:

```
1 if v noch kein Follower then
2     v sendet ID(v) an Nachbarn im und gegen den Uhrzeigersinn
```

Jede andere Runde:

```
1 if v empfängt Nachricht M von Nachbar gegen (im) den Uhrzeigersinn then
2     if M < ID(v) then
3         v wird zum Follower (sofern nicht bereits vorher geschehen)
4         if nicht letzte Runde der Phase then
5             v sendet M an Nachbar im (gegen) Uhrzeigersinn
```

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i-te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v:

Erste Runde jeder Phase:

```
1 if v noch kein Follower then
2     v sendet ID(v) an Nachbarn im und gegen den Uhrzeigersinn
```

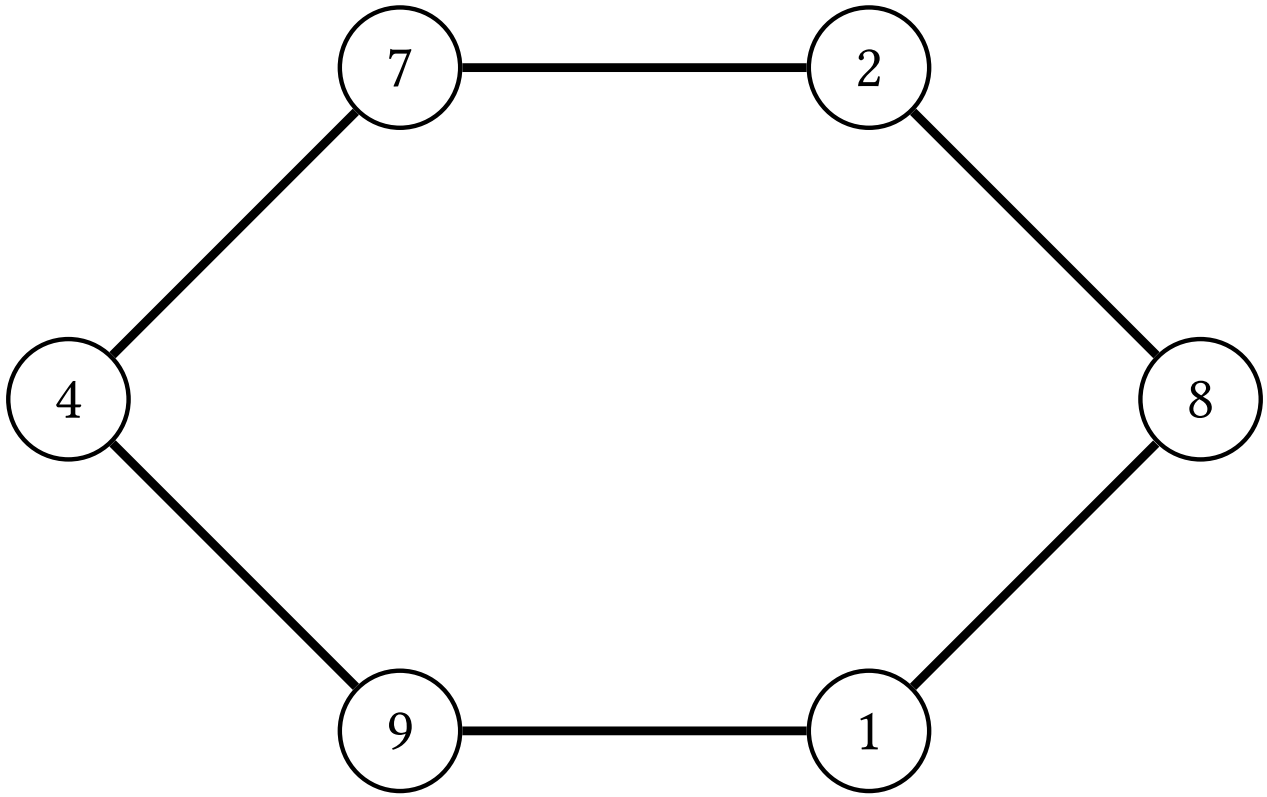
Jede andere Runde:

```
1 if v empfängt Nachricht M von Nachbar gegen (im) den Uhrzeigersinn then
2     if M < ID(v) then
3         v wird zum Follower (sofern nicht bereits vorher geschehen)
4         if nicht letzte Runde der Phase then
5             v sendet M an Nachbar im (gegen) Uhrzeigersinn
```

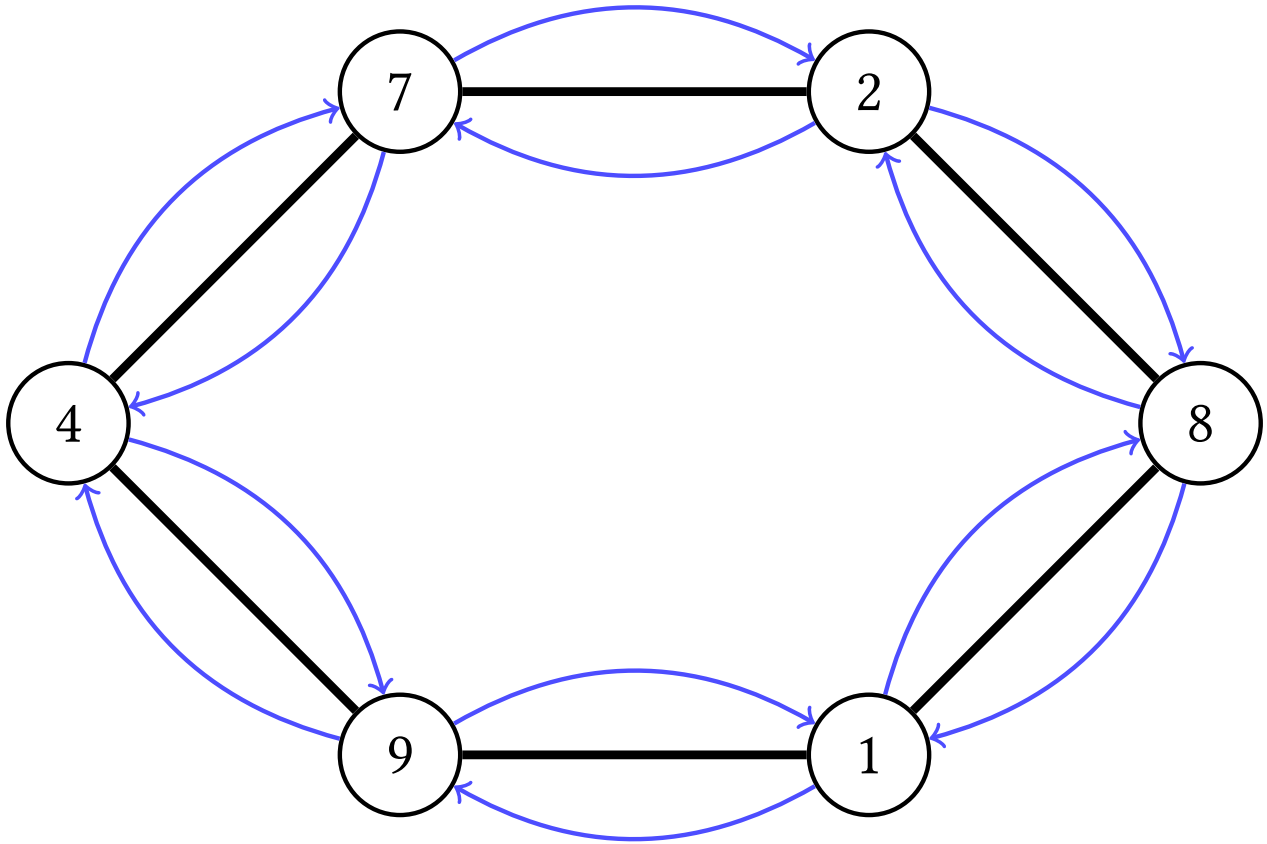
Zusätzlich in letzter Runde der letzten Phase:

```
1 if v ist noch kein Follower then  v wird zum Leader
```

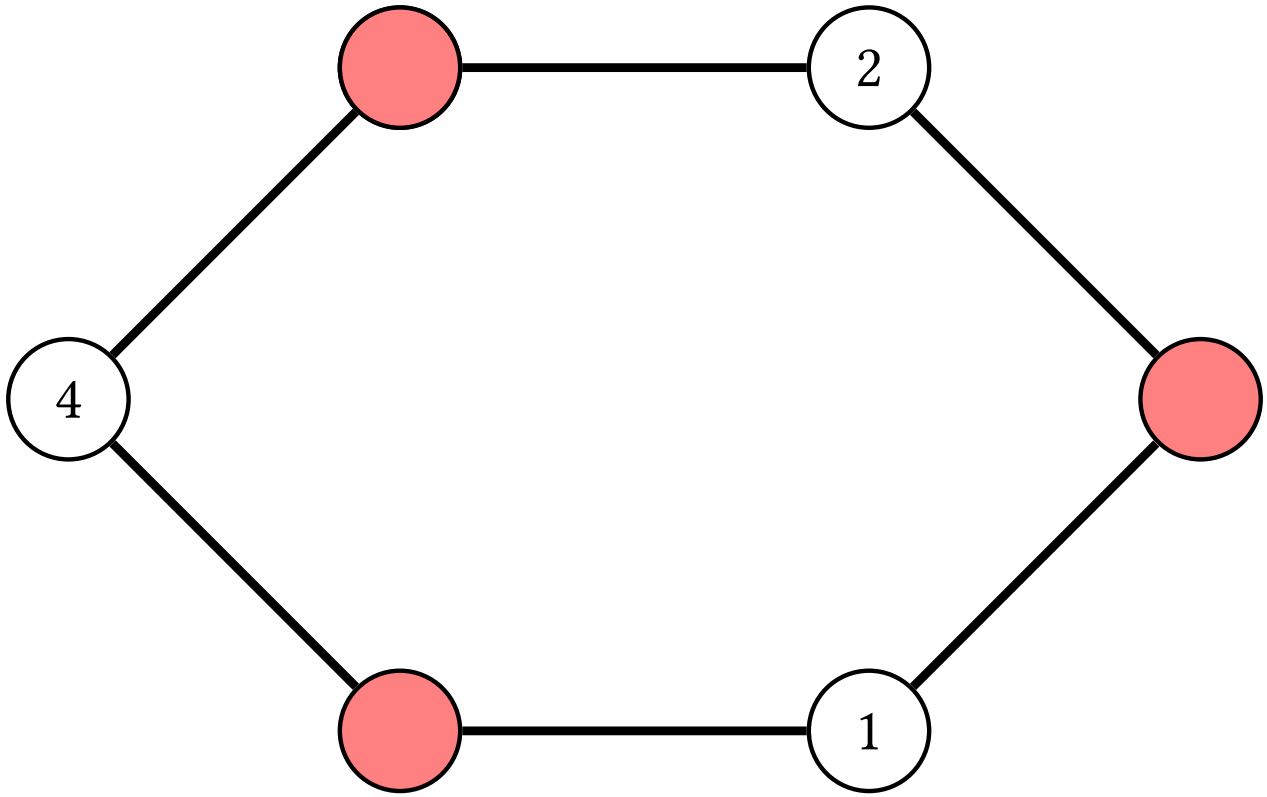
Beispiel



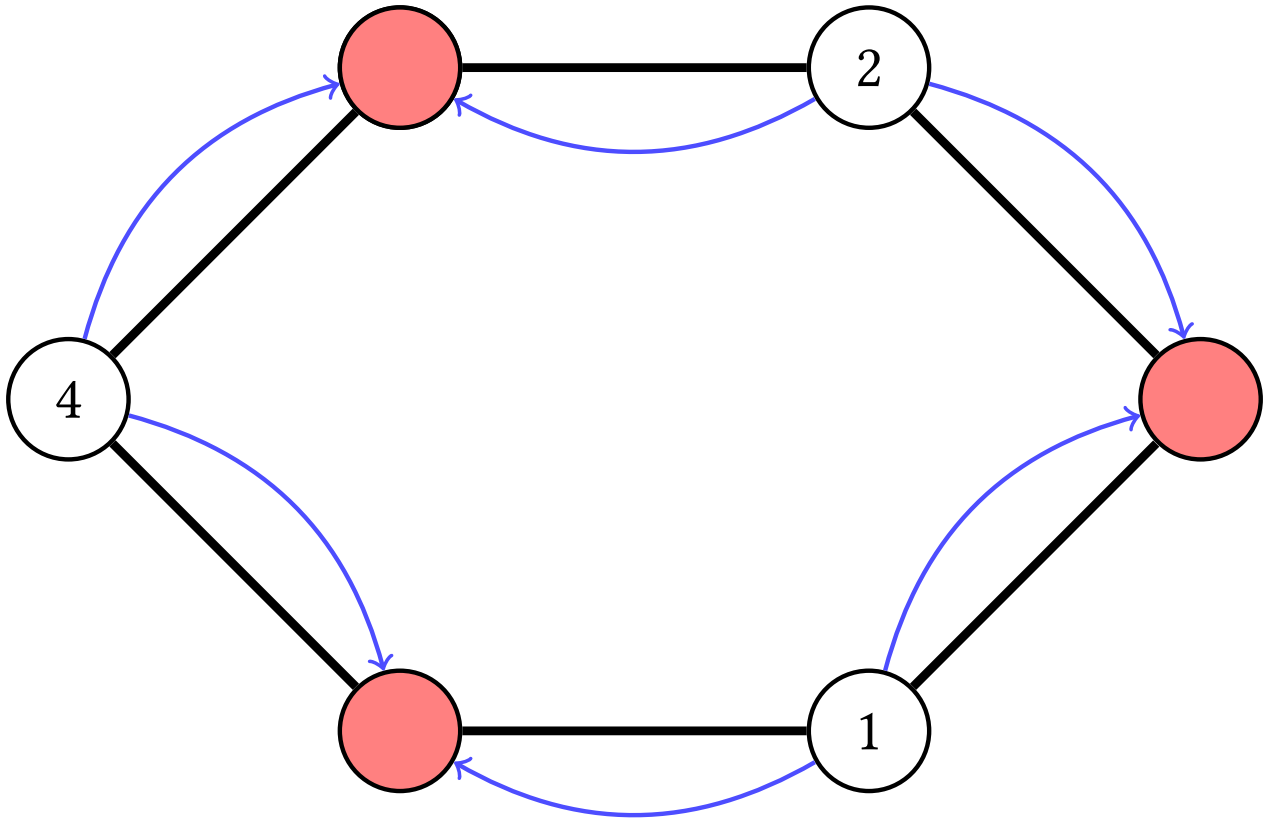
Beispiel



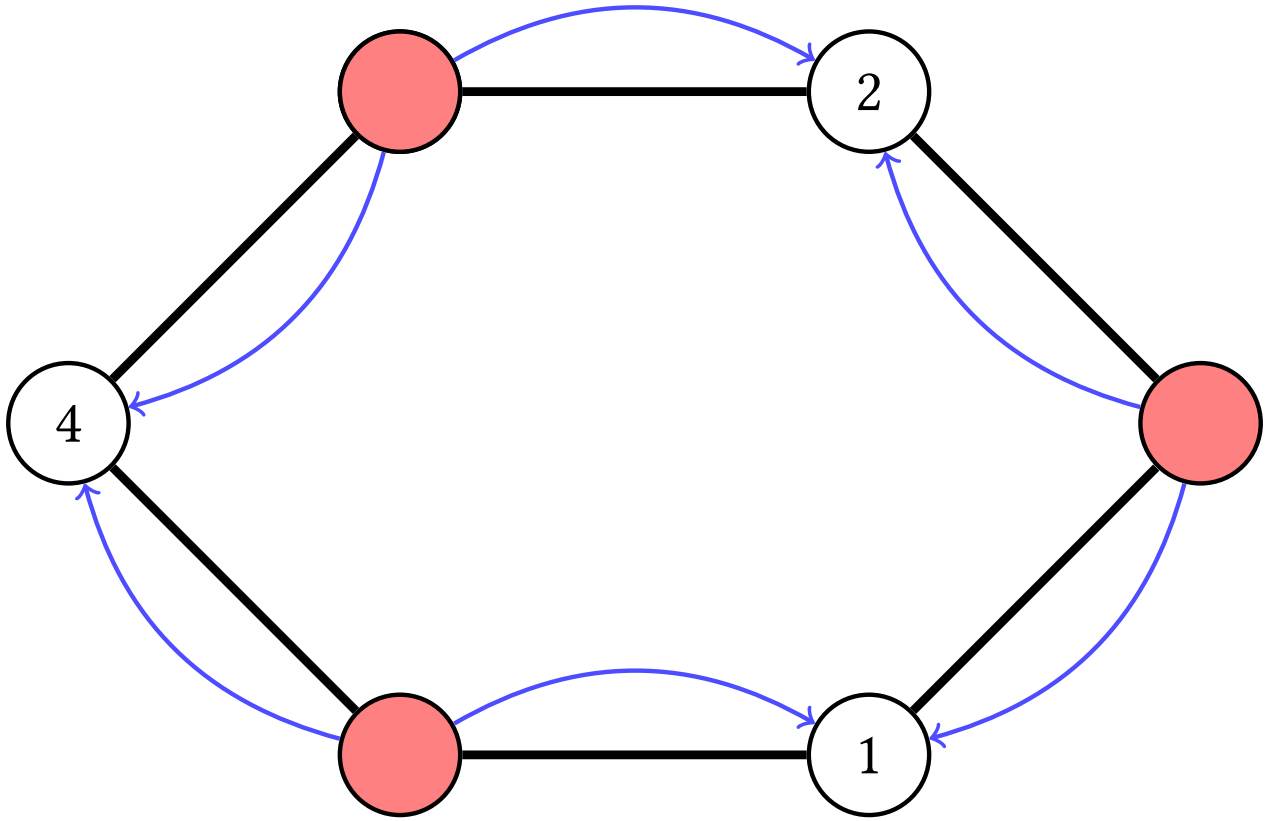
Beispiel



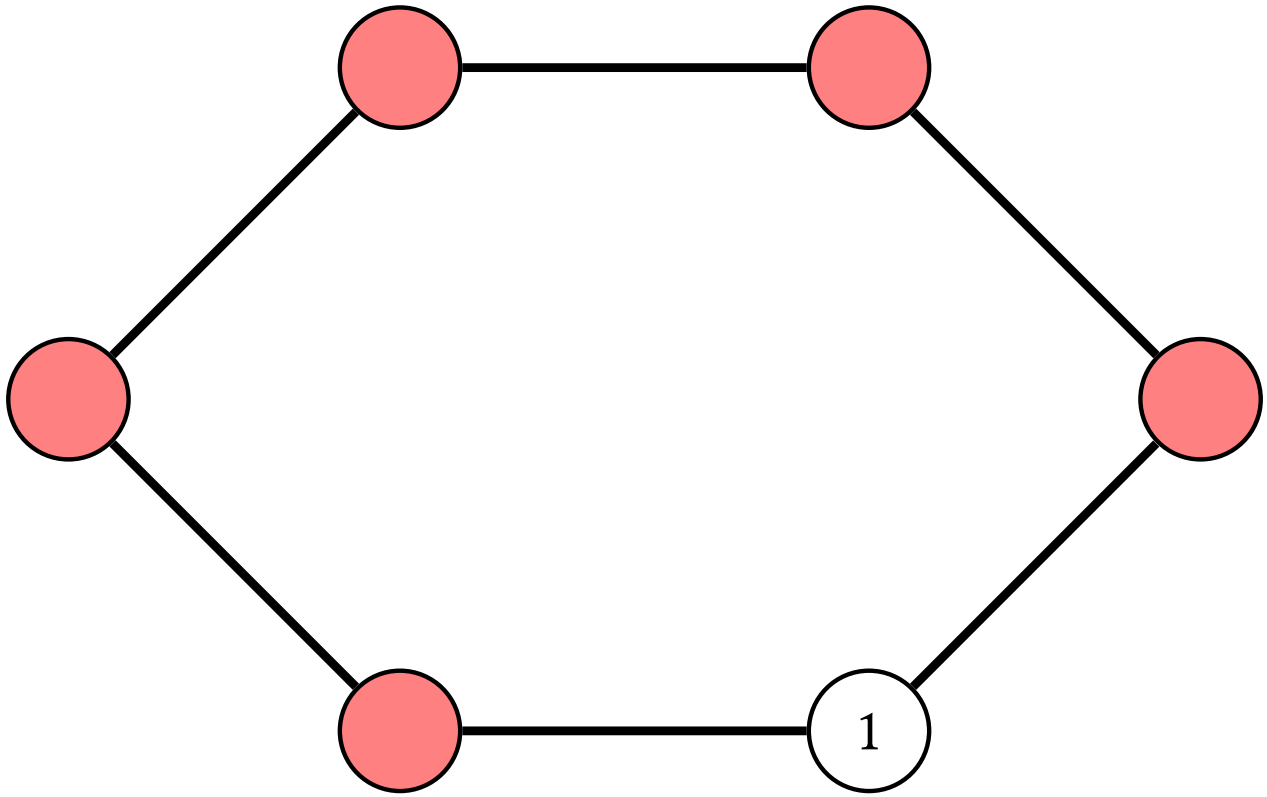
Beispiel



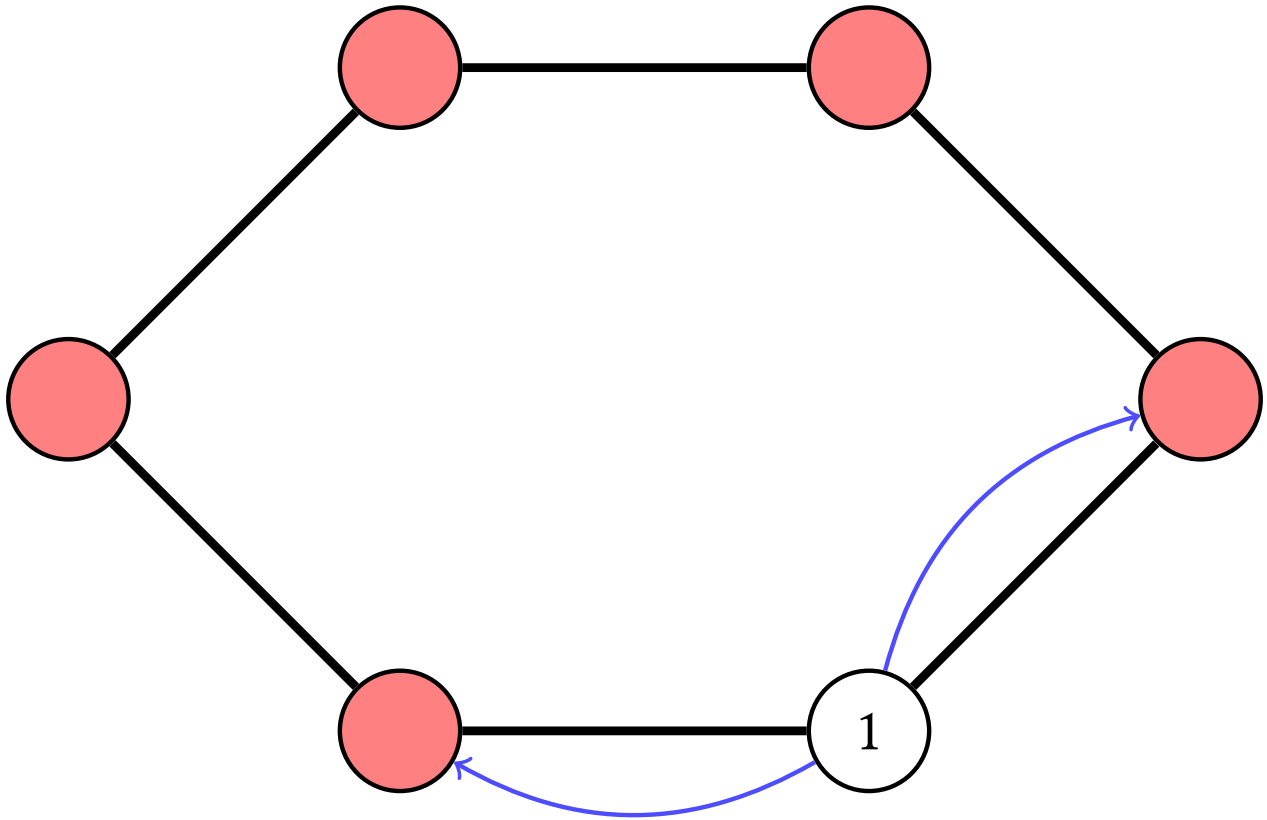
Beispiel



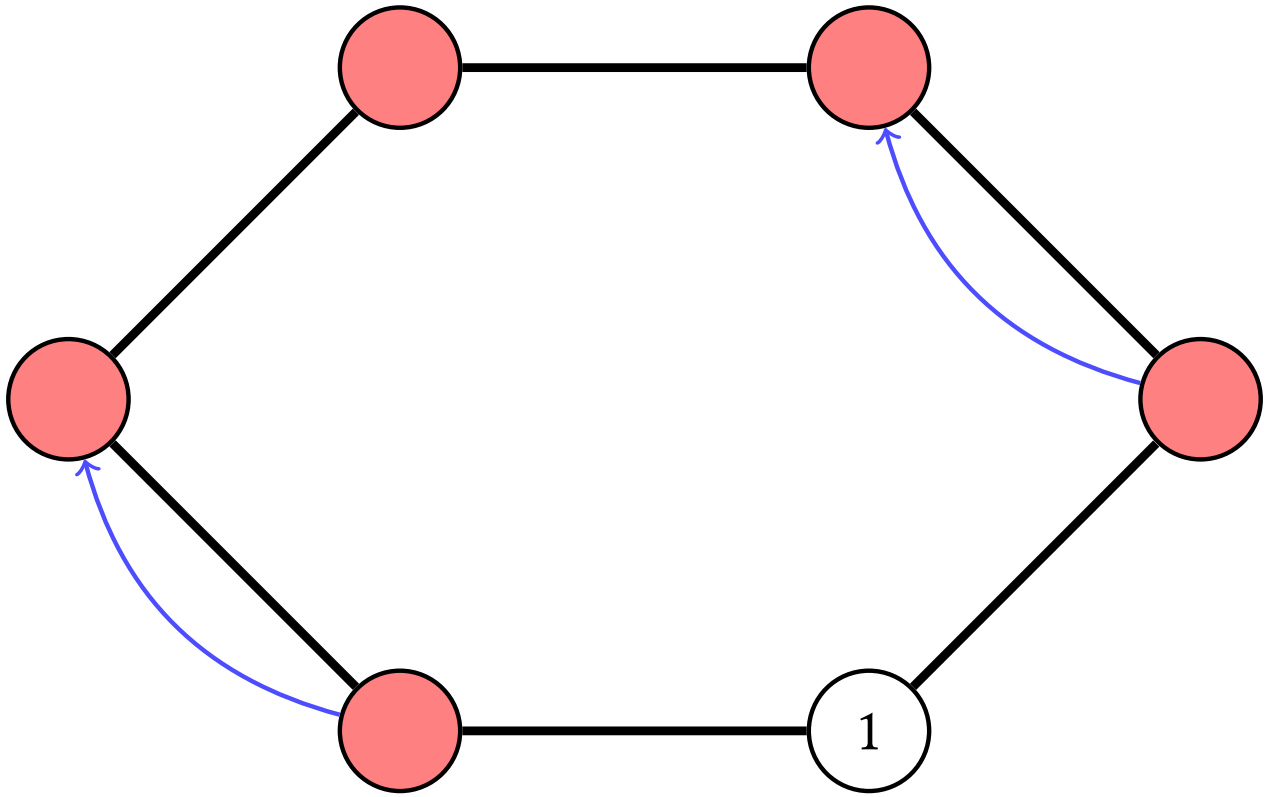
Beispiel



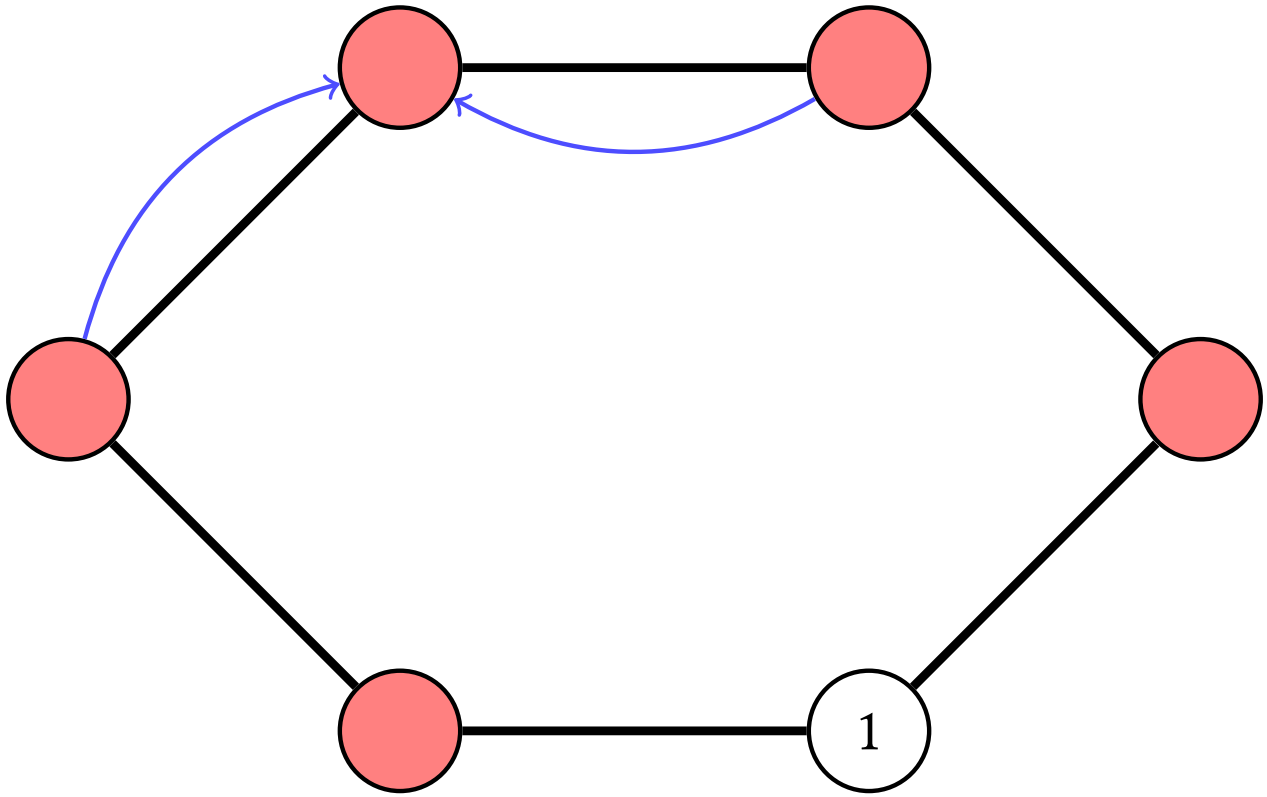
Beispiel



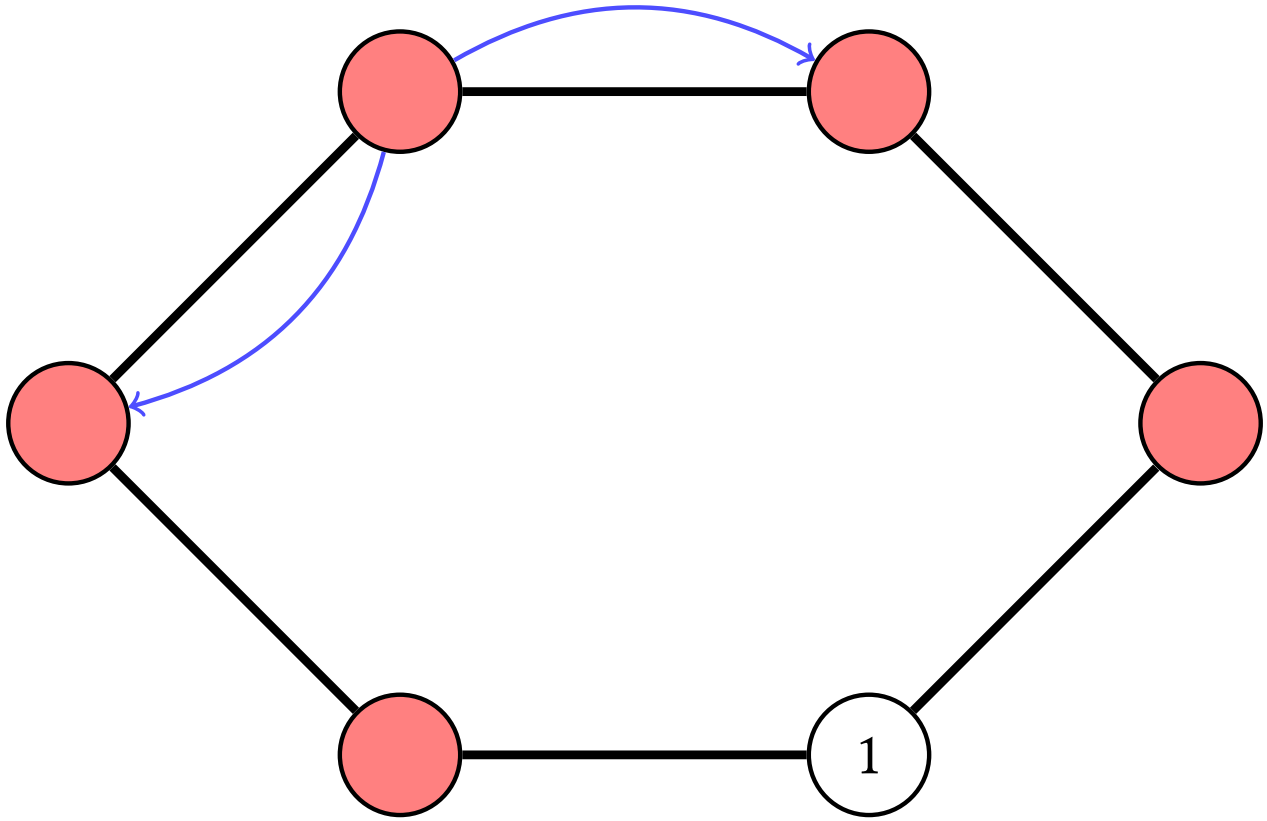
Beispiel



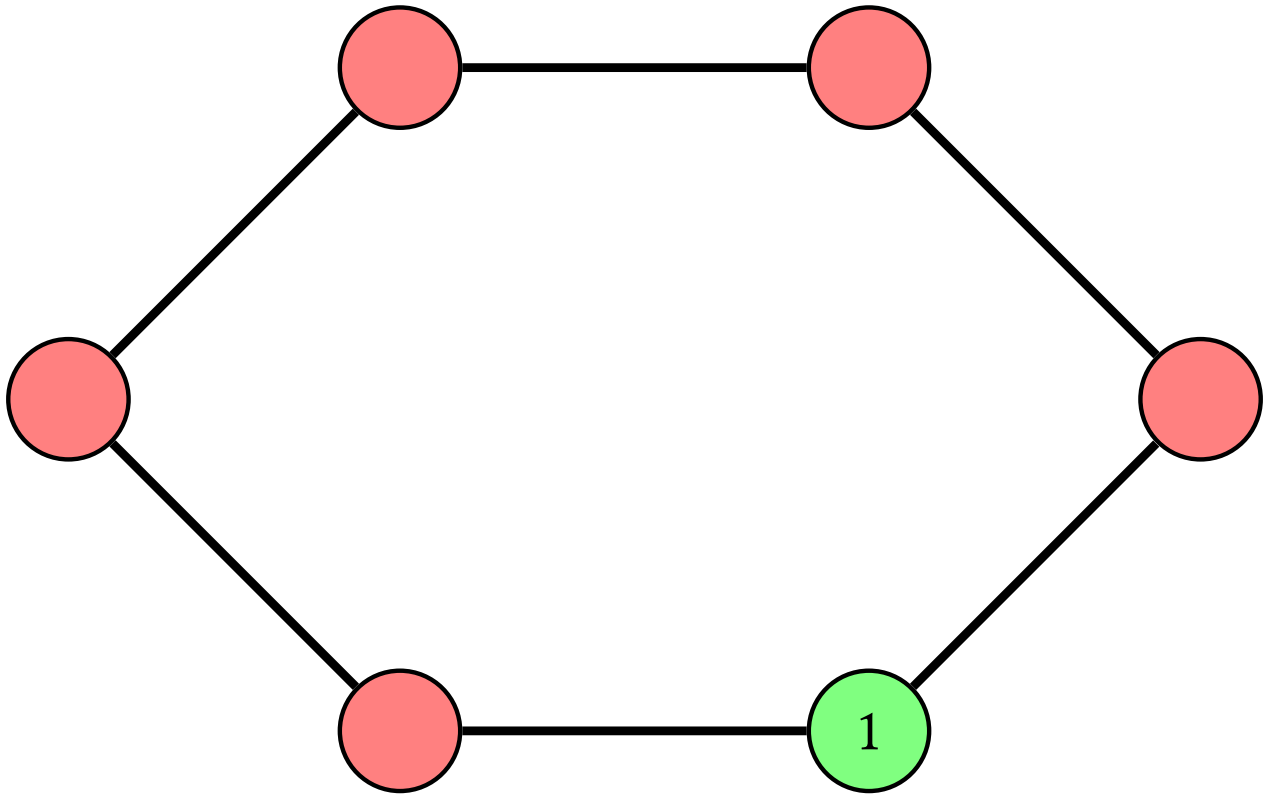
Beispiel



Beispiel



Beispiel



Korrektheit

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Korrektheit

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzer Phase zum Leader

Korrektheit

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzer Phase zum Leader
- Nachricht mit ID von z in letzer Phase ℓ (wobei $\ell = \lceil \log n \rceil$) erreicht $2^{\ell-1}$ Knoten jeweils im und gegen den Uhrzeigersinn

Korrektheit

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzer Phase zum Leader
- Nachricht mit ID von z in letzer Phase ℓ (wobei $\ell = \lceil \log n \rceil$) erreicht $2^{\ell-1}$ Knoten jeweils im und gegen den Uhrzeigersinn
- Da $2^{\ell-1} = 2^{\lceil \log n \rceil - 1} \geq 2^{\log n - 1} = \frac{2^{\log n}}{2} = \frac{n}{2}$, erreicht Nachricht mit ID von z alle anderen Knoten, die spätestens dann zu Followern werden, da sie höhere ID als z haben

Korrektheit

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzer Phase zum Leader
- Nachricht mit ID von z in letzer Phase ℓ (wobei $\ell = \lceil \log n \rceil$) erreicht $2^{\ell-1}$ Knoten jeweils im und gegen den Uhrzeigersinn
- Da $2^{\ell-1} = 2^{\lceil \log n \rceil - 1} \geq 2^{\log n - 1} = \frac{2^{\log n}}{2} = \frac{n}{2}$, erreicht Nachricht mit ID von z alle anderen Knoten, die spätestens dann zu Followern werden, da sie höhere ID als z haben
- Somit: Gültige Einteilung in Leader und Follower

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i$$

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1}$$

Geometrische Reihe: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2}$$

Geometrische Reihe: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2} \leq 4 \cdot 2^{\log n}$$

Geometrische Reihe: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2} \leq 4 \cdot 2^{\log n} = 4n$$

Geometrische Reihe: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i: alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i: alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1}, der am Ende von Phase i aktiv ist

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i: alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1}, der am Ende von Phase i aktiv ist
- Dann war w auch am Anfang von Phase i aktiv

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i: alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1}, der am Ende von Phase i aktiv ist
- Dann war w auch am Anfang von Phase i aktiv
- Wenn $\mathrm{ID}(v) > \mathrm{ID}(w)$: v erhält in Phase i Nachricht mit $\mathrm{ID}(w)$ (da w am Anfang der Phase aktiv war) und wird daher inaktiv: Widerspruch zur Annahme, dass v am Ende von Phase i aktiv ist

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i: alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1}, der am Ende von Phase i aktiv ist
- Dann war w auch am Anfang von Phase i aktiv
- Wenn $\mathrm{ID}(v) > \mathrm{ID}(w)$: v erhält in Phase i Nachricht mit $\mathrm{ID}(w)$ (da w am Anfang der Phase aktiv war) und wird daher inaktiv: Widerspruch zur Annahme, dass v am Ende von Phase i aktiv ist
- Wenn $\mathrm{ID}(v) < \mathrm{ID}(w)$: w erhält in Phase i Nachricht mit $\mathrm{ID}(v)$ und wird daher inaktiv: Widerspruch zur Annahme, dass w am Ende von Phase i aktiv ist

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

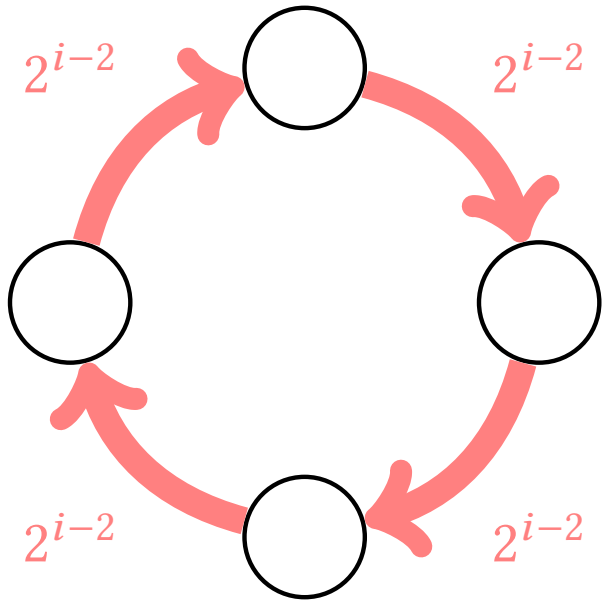
- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden

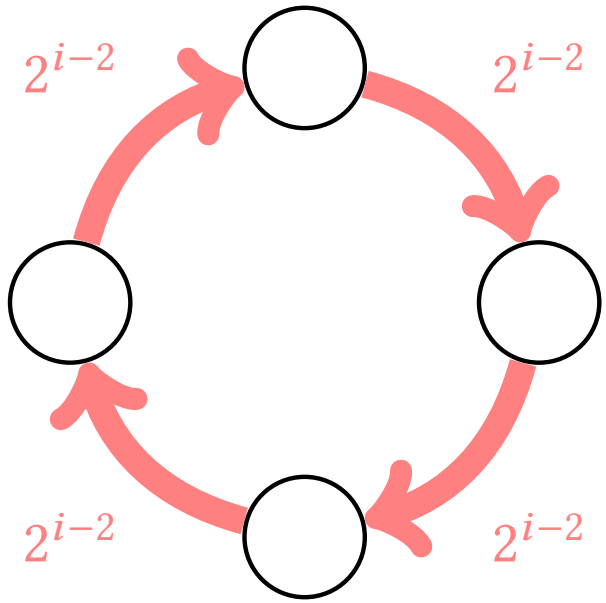


Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden



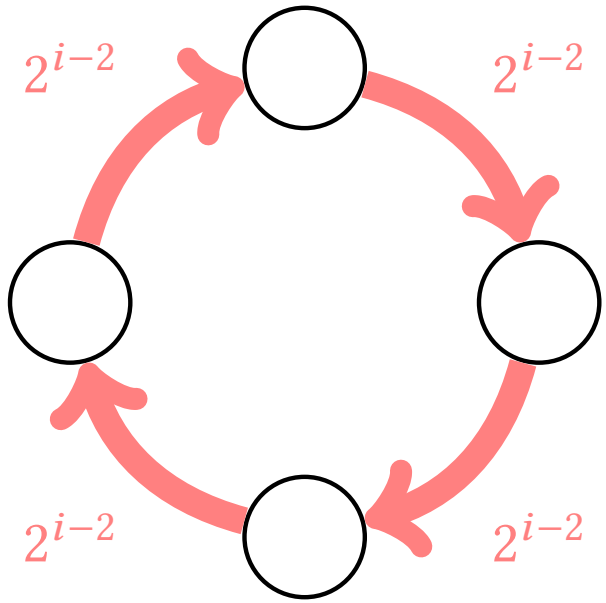
- Sei a die Anzahl aktiver Knoten

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden



- Sei a die Anzahl aktiver Knoten
- Dann gilt $a \cdot 2^{i-2} \leq n$ und somit $a \leq n/2^{i-2}$

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet
- Anzahl an Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet
- Anzahl an Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$
- Anzahl an Nachrichten in Phase 1: $2n$

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet
- Anzahl an Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$
- Anzahl an Nachrichten in Phase 1: $2n$
- $\lceil \log n \rceil$ Phasen, daher insgesamt $O(n \log n)$ Nachrichten

Zusammenfassung

Radius Growth:

- Bestimmt Knoten mit kleinster ID zum Leader
- Laufzeit: $O(n)$
- Nachrichtenkomplexität: $O(n \log n)$
- Kann auch als asynchroner Algorithmus in uniformen Ringen formuliert werden

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

1. Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
2. Knoten führen Clockwise Algorithmus mit gewählten IDs aus

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

1. Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
2. Knoten führen Clockwise Algorithmus mit gewählten IDs aus
   Problem:
   - IDs können mehrfach vergeben sein
   - Algorithmus könnte falsches Ergebnis liefern
   - Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

1. Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
2. Knoten führen Clockwise Algorithmus mit gewählten IDs aus
   Problem:
   - IDs können mehrfach vergeben sein
   - Algorithmus könnte falsches Ergebnis liefern
   - Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern
3. Knoten verifizieren, ob Ergebnis korrekt ist

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

1. Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
2. Knoten führen Clockwise Algorithmus mit gewählten IDs aus
   Problem:
   - IDs können mehrfach vergeben sein
   - Algorithmus könnte falsches Ergebnis liefern
   - Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern
3. Knoten verifizieren, ob Ergebnis korrekt ist

   Gültigkeit einer Einteilung in Leader und Follower kann in $O(n)$ Runden mit $O(n)$ Nachrichten überprüft werden

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

1. Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
2. Knoten führen Clockwise Algorithmus mit gewählten IDs aus
   Problem:
   - IDs können mehrfach vergeben sein
   - Algorithmus könnte falsches Ergebnis liefern
   - Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern
3. Knoten verifizieren, ob Ergebnis korrekt ist
   Gültigkeit einer Einteilung in Leader und Follower kann in $O(n)$ Runden mit $O(n)$ Nachrichten überprüft werden
4. Falls Ergebnis nicht korrekt, wiederhole ab Schritt 1

Erfolgswahrscheinlichkeit einer Iteration I

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Erfolgswahrscheinlichkeit einer Iteration I

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration jedenfalls erfolgreich.

Erfolgswahrscheinlichkeit einer Iteration I

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration jedenfalls erfolgreich.

- Zufallsvariablen X_v (für alle $v \in V$):

$$X_v = \begin{cases} 1 & \text{falls Knoten } v \text{ ID } 0 \text{ erhält} \\ 0 & \text{ansonsten} \end{cases}$$

Erfolgswahrscheinlichkeit einer Iteration I

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration jedenfalls erfolgreich.

- Zufallsvariablen X_v (für alle $v \in V$):

$$X_v = \begin{cases} 1 & \text{falls Knoten } v \text{ ID } 0 \text{ erhält} \\ 0 & \text{ansonsten} \end{cases}$$

- $\Pr[X_v = 1] = \frac{1}{2n}$, da ID uniform aus $0 \ldots 2n-1$ gewählt wird

Erfolgswahrscheinlichkeit einer Iteration I

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration jedenfalls erfolgreich.

- Zufallsvariablen X_v (für alle $v \in V$):

$$X_v = \begin{cases} 1 & \text{falls Knoten } v \text{ ID } 0 \text{ erhält} \\ 0 & \text{ansonsten} \end{cases}$$

- $\Pr[X_v = 1] = \frac{1}{2n}$, da ID uniform aus $0 \ldots 2n-1$ gewählt wird
- Jedes X_v ist Bernoulli-verteilt mit Erfolgswahrscheinlichkeit $p := \frac{1}{2n}$

Erfolgswahrscheinlichkeit einer Iteration I

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration jedenfalls erfolgreich.

- Zufallsvariablen X_v (für alle $v \in V$):

$$X_v = \begin{cases} 1 & \text{falls Knoten } v \text{ ID } 0 \text{ erhält} \\ 0 & \text{ansonsten} \end{cases}$$

- $\Pr[X_v = 1] = \frac{1}{2n}$, da ID uniform aus $0 \ldots 2n-1$ gewählt wird
- Jedes X_v ist Bernoulli-verteilt mit Erfolgswahrscheinlichkeit $p := \frac{1}{2n}$
- Zufallsvariable $X := \sum_{v \in V} X_v$ für Anzahl an Knoten mit ID 0

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1}$$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1}$$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1 + xn$ für $x \geq -1$ und $n \geq 0$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$
$$\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right)$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1 + xn$ für $x \geq -1$ und $n \geq 0$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$
$$\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) = \frac{1}{4}$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1 + xn$ für $x \geq -1$ und $n \geq 0$

Erfolgswahrscheinlichkeit einer Iteration II

Beobachtung

X_v's sind stochastisch unabhängig und $X := \sum_{v \in V} X_v$ ist somit binomialverteilt (mit Parametern n und p).

Daher gilt: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ für alle $0 \leq k \leq n$

Insbesondere: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$
$$\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) = \frac{1}{4}$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1 + xn$ für $x \geq -1$ und $n \geq 0$

Somit: Jede Iteration mit Wahrscheinlichkeit $q \geq \frac{1}{4}$ erfolgreich

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k \geq 1} A_k$: Algorithmus terminiert irgendwann

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k \geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k \geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k \geq 1} A_k\right] = \sum_{k \geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q$$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k \geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k \geq 1} A_k\right] = \sum_{k \geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k\geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k}$$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k\geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)}$$

Geometrische Reihe: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k\geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)} = q \cdot \frac{1}{q}$$

Geometrische Reihe: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k\geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)} = q \cdot \frac{1}{q} = 1$$

Geometrische Reihe: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Ereignis A_k: Algorithmus terminiert nach (genau) k Iterationen
- Ereignis $\bigcup_{k\geq 1} A_k$: Algorithmus terminiert irgendwann
- Wegen Disjunktheit der A_k's:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)} = q \cdot \frac{1}{q} = 1$$

Geometrische Reihe: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Achtung: „Wahrscheinlichkeit 1“ ≠ „immer“ – Gegenereignis ist möglich!

Geometrische Verteilung

Zufallsvariablen Y_i (für $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{falls } i\text{-te Iteration des Algorithmus erfolgreich ist} \\ 0 & \text{ansonsten} \end{cases}$$

Geometrische Verteilung

Zufallsvariablen Y_i (für $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{falls } i\text{-te Iteration des Algorithmus erfolgreich ist} \\ 0 & \text{ansonsten} \end{cases}$$

Zufallsvariable $Z = \min\{i \geq 1 \mid Y_i = 1\}$ für Anzahl an Iterationen des Algorithmus

Geometrische Verteilung

Zufallsvariablen Y_i (für $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{falls } i\text{-te Iteration des Algorithmus erfolgreich ist} \\ 0 & \text{ansonsten} \end{cases}$$

Zufallsvariable $Z = \min\{i \geq 1 \mid Y_i = 1\}$ für Anzahl an Iterationen des Algorithmus

Beobachtung

Y_i's sind stochastisch unabhängig und Z ist somit geometrisch verteilt (mit Parameter q).

Geometrische Verteilung

Zufallsvariablen Y_i (für $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{falls } i\text{-te Iteration des Algorithmus erfolgreich ist} \\ 0 & \text{ansonsten} \end{cases}$$

Zufallsvariable $Z = \min\{i \geq 1 \mid Y_i = 1\}$ für Anzahl an Iterationen des Algorithmus

Beobachtung

Y_i's sind stochastisch unabhängig und Z ist somit geometrisch verteilt (mit Parameter q).

Daher gilt: $\Pr[Z = k] = q(1-q)^{k-1}$ für alle $k \geq 1$

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in jeder Iteration

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in jeder Iteration
- Mit Erwartungswert geometrisch verteilter Zufallsvariable Z: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in jeder Iteration
- Mit Erwartungswert geometrisch verteilter Zufallsvariable Z: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Frage: Erwartungswert erlaubt Ausreißer nach oben; ab wie vielen Iterationen kann man sich „ziemlich sicher“ sein, dass der Algorithmus terminiert?

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in jeder Iteration
- Mit Erwartungswert geometrisch verteilter Zufallsvariable Z: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Frage: Erwartungswert erlaubt Ausreißer nach oben; ab wie vielen Iterationen kann man sich „ziemlich sicher“ sein, dass der Algorithmus terminiert?

Definition

Ein Ereignis findet **mit hoher Wahrscheinlichkeit** statt, wenn es mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$, für eine beliebig vorgegebene Konstante $c \geq 1$, stattfindet.

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in jeder Iteration
- Mit Erwartungswert geometrisch verteilter Zufallsvariable Z: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Frage: Erwartungswert erlaubt Ausreißer nach oben; ab wie vielen Iterationen kann man sich „ziemlich sicher“ sein, dass der Algorithmus terminiert?

Definition

Ein Ereignis findet **mit hoher Wahrscheinlichkeit** statt, wenn es mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$, für eine beliebig vorgegebene Konstante $c \geq 1$, stattfindet. (Insbesondere gilt: $\lim_{n\to\infty} 1 - \frac{1}{n^c} = 1$)

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1 - q)^k$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c}$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}}$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}} = \frac{1}{n^c}$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$
$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}} = \frac{1}{n^c}$$

- **Somit:** $\Pr[Z \leq k] = 1 - \Pr[Z > k] \geq 1 - \frac{1}{n^c}$

Zusammenfassung

Randomisierter Algorithmus :

- Wählt IDs zufällig
- Macht Knoten mit kleinster ID zum Leader
- Wiederholung bis Leader Election erfolgreich
- Laufzeit: $O(n)$ Runden in Erwartung
- Nachrichtenkomplexität: $O(n^2)$ in Erwartung Bei naiver Analyse mit Clockwise Algorithmus
- Kann auch als asynchroner Algorithmus formuliert werden

Allgemeines Prinzip

Wir haben implizit gezeigt:

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo → Las Vegas)

- *Sei $\mathcal{A}$ ein randomisierter Algorithmus für ein Problem $\mathcal{P}$, der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.*

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo $\rightarrow$ Las Vegas)

- *Sei $\mathcal{A}$ ein randomisierter Algorithmus für ein Problem $\mathcal{P}$, der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.*
- *Sei $\mathcal{V}$ ein Verifizierer, der für jede Ausgabe von $\mathcal{A}$ ermitteln kann, ob die Lösung korrekt ist.*

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo → Las Vegas)

- *Sei $\mathcal{A}$ ein randomisierter Algorithmus für ein Problem $\mathcal{P}$, der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.*
- *Sei $\mathcal{V}$ ein Verifizierer, der für jede Ausgabe von $\mathcal{A}$ ermitteln kann, ob die Lösung korrekt ist.*
- *Dann gibt es einen Algorithmus $\mathcal{B}$, der mit Wahrscheinlichkeit 1 eine korrekte Lösung berechnet und dafür $\mathcal{A}$ und $\mathcal{V}$ in Erwartung $O(1/q)$ Mal aufruft (oder: $O(c \log_{1-1/q} n)$ Mal mit Wahrscheinlichkeit $1 - \frac{1}{n^c}$).*

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo → Las Vegas)

- *Sei $\mathcal{A}$ ein randomisierter Algorithmus für ein Problem $\mathcal{P}$, der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.*
- *Sei $\mathcal{V}$ ein Verifizierer, der für jede Ausgabe von $\mathcal{A}$ ermitteln kann, ob die Lösung korrekt ist.*
- *Dann gibt es einen Algorithmus $\mathcal{B}$, der mit Wahrscheinlichkeit 1 eine korrekte Lösung berechnet und dafür $\mathcal{A}$ und $\mathcal{V}$ in Erwartung $O(1/q)$ Mal aufruft (oder: $O(c \log_{1-1/q} n)$ Mal mit Wahrscheinlichkeit $1 - \frac{1}{n^c}$).*

Definition

Ein randomisierter Algorithmus $\mathcal{B}$, dessen Ergebnis immer korrekt ist, heißt **Las Vegas** Algorithmus.

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo → Las Vegas)

- *Sei $\mathcal{A}$ ein randomisierter Algorithmus für ein Problem $\mathcal{P}$, der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.*
- *Sei $\mathcal{V}$ ein Verifizierer, der für jede Ausgabe von $\mathcal{A}$ ermitteln kann, ob die Lösung korrekt ist.*
- *Dann gibt es einen Algorithmus $\mathcal{B}$, der mit Wahrscheinlichkeit 1 eine korrekte Lösung berechnet und dafür $\mathcal{A}$ und $\mathcal{V}$ in Erwartung $O(1/q)$ Mal aufruft (oder: $O(c \log_{1-1/q} n)$ Mal mit Wahrscheinlichkeit $1 - \frac{1}{n^c}$).*

Definition

Ein randomisierter Algorithmus $\mathcal{B}$, dessen Ergebnis immer korrekt ist, heißt **Las Vegas** Algorithmus. Ein randomisierter Algorithmus $\mathcal{A}$, dessen Ergebnis wahrscheinlich korrekt ist, heißt **Monte Carlo** Algorithmus.

Zusammenfassung

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

Zusammenfassung

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform

Zusammenfassung

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen

Zusammenfassung

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen
- Komplexitätsmaße: Zeit, #Nachrichten

Zusammenfassung

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen
- Komplexitätsmaße: Zeit, #Nachrichten
- Obere und untere Schranken (bzw. Unmöglichkeit)

Zusammenfassung

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen
- Komplexitätsmaße: Zeit, #Nachrichten
- Obere und untere Schranken (bzw. Unmöglichkeit)
- Zwei Möglichkeiten Symmetrien zu brechen:
  - Eindeutige IDs
  - Randomisierung

Quellen

Der Inhalt dieser Vorlesungseinheit basiert zum Teil auf Vorlesungseinheiten von Robert Elsässer und Stefan Schmid.

Literatur:

- Hagit Attiya, Jennifer Welch (2004) *Distributed Computing*, Kapitel 3 u. 14, Wiley.
- Daniel S. Hirschberg, James B. Sinclair. „Decentralized Extrema-Finding in Circular Configurations of Processors“. *Communications of the ACM* 23(11): 627–628 (1980)
- Alon Itai, Michael Rodeh. „Symmetry breaking in distributed networks“. *Information and Computation* 88(1): 60–87 (1990)
- Nancy A. Lynch (1996) *Distributed Algorithms*, Kapitel 3, Morgan Kaufmann.