Leader Election I

Algorithmen für verteilte Systeme

Sebastian Forster

Universität Salzburg

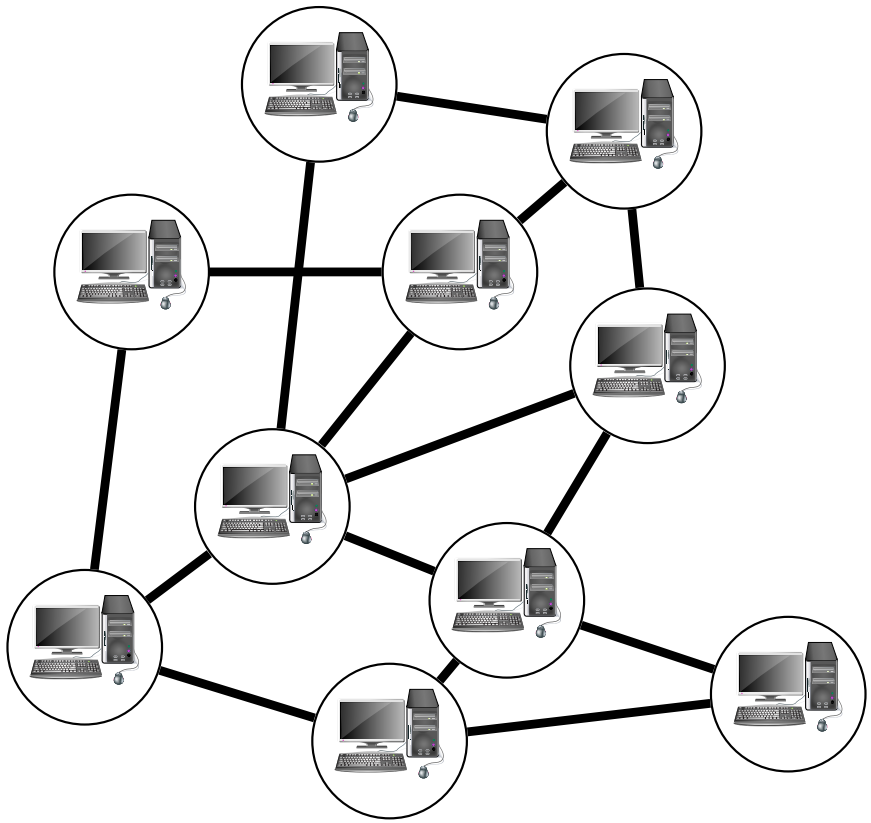
Prozessornetzwerke

Ausgangssituation: Mehrere gleichartige Prozessoren, die über Direktverbindungen kommunizieren

Prozessornetzwerke

Ausgangssituation: Mehrere gleichartige Prozessoren, die über Direktverbindungen kommunizieren

- Prozessornetzwerke werden als ungerichtete Graphen modelliert.
- Jedem Knoten des Graphen entspricht ein Prozessor mit lokalem Speicher und jeder Kante ein Kommunikationskanal.



Netzwerke als Graphen

Definition

Ein **Graph** $G = (V, E)$ ist ein Paar bestehend aus einer Menge von Knoten V und einer Menge von Kanten E, wobei

- in einem **gerichteten** Graph jede Kante $e = (u, v)$ in (geordnetes) Paar zweier Knoten ist und
- in einem **ungerichteten** Graph jede Kante $e = \{u, v\}$ eine Menge zweier Knoten ist.

Anmerkung: Anstelle von $\{u, v\}$ schreiben wir dennoch oft (u, v)

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Followers*

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Followers*
- **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 *Followers*
- **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

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Followers*
- **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

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus aktiv, wenn er eine Nachricht empfängt

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus aktiv, wenn er eine Nachricht empfängt
  - Jede Nachrichtenübermittlung benötigt endliche Zeit

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus aktiv, wenn er eine Nachricht empfängt
  - Jede Nachrichtenübermittlung benötigt endliche Zeit
  - Für Laufzeitanalyse: Delay von höchstens einer Zeiteinheit

Kommunikationsmodell

- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus aktiv, wenn er eine Nachricht empfängt
  - Jede Nachrichtenübermittlung benötigt endliche Zeit
  - Für Laufzeitanalyse: Delay von höchstens einer Zeiteinheit
- Komplexitätsmaße: Anzahl Zeiteinheiten (Runden) und Anzahl gesendeter Nachrichten (ohne interne Berechnungszeit)

Varianten

- Synchron vs. asynchron

Varianten

- Synchron vs. asynchron
- Anonym vs. Knoten besitzen eindeutige IDs (UIDs)

Varianten

- Synchron vs. asynchron
- Anonym vs. Knoten besitzen eindeutige IDs (UIDs)
  IDs sind in der Regel Bit-Strings, oft der Länge $O(\log n)$
  Können als nichtnegative ganze Zahlen interpretiert werden

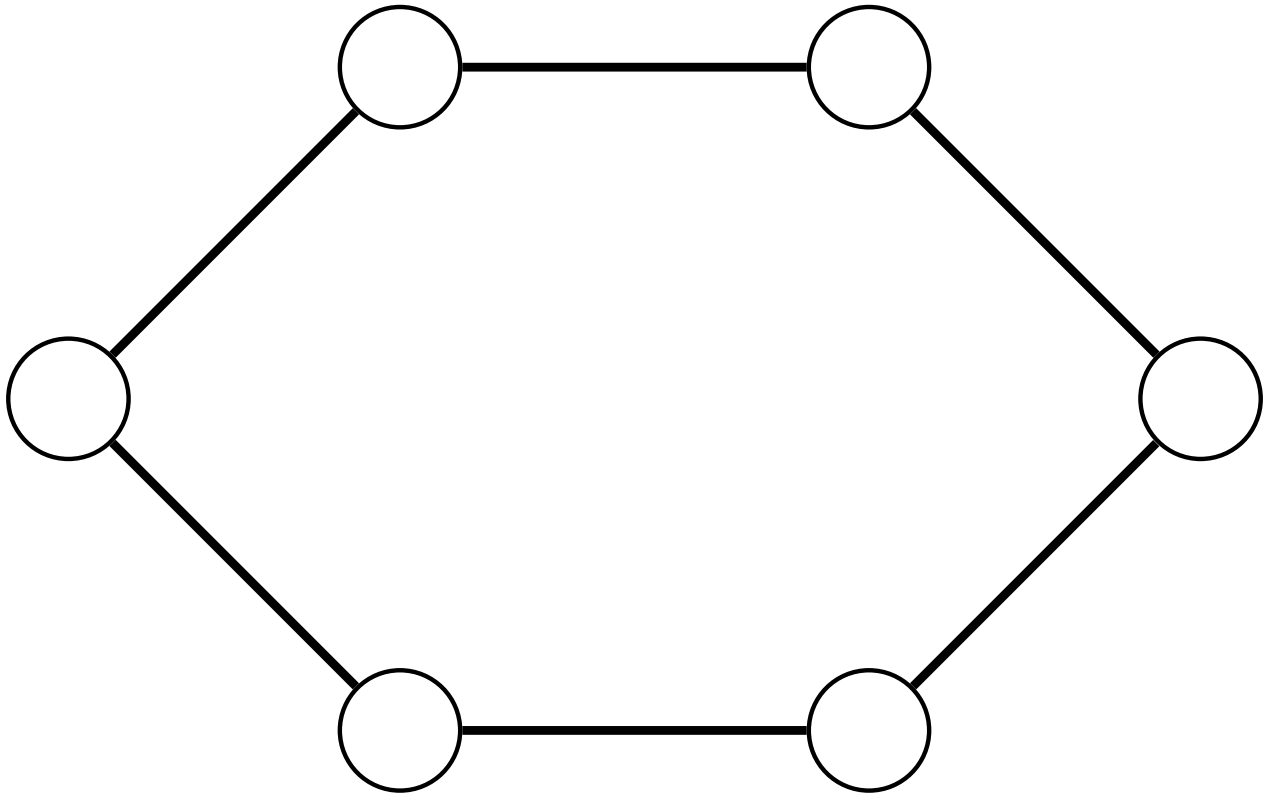
Varianten

- Synchron vs. asynchron
- Anonym vs. Knoten besitzen eindeutige IDs (UIDs)
  IDs sind in der Regel Bit-Strings, oft der Länge $O(\log n)$
  Können als nichtnegative ganze Zahlen interpretiert werden
- Uniform vs. Anzahl der Knoten n ist globales Wissen

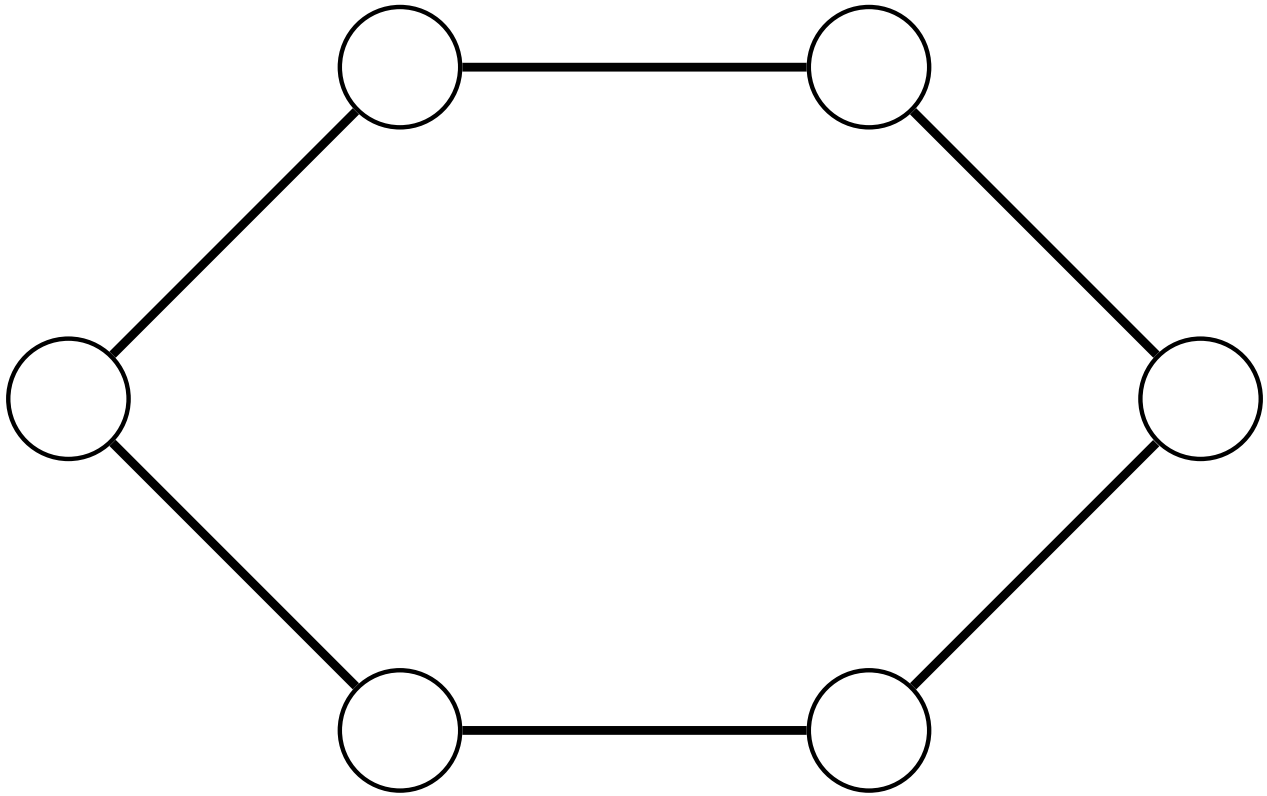
Varianten

- Synchron vs. asynchron
- Anonym vs. Knoten besitzen eindeutige IDs (UIDs)
  IDs sind in der Regel Bit-Strings, oft der Länge $O(\log n)$
  Können als nichtnegative ganze Zahlen interpretiert werden
- Uniform vs. Anzahl der Knoten n ist globales Wissen
  Manchmal ist Approximation $\hat{n}$ des exakten Werts n ausreichend

Ziel: Leader im Ring



Ziel: Leader im Ring



Jeder Knoten hat zwei Ports zu Nachbarknoten:

- „Im Uhrzeigersinn“ (Clockwise)
- „Gegen den Uhrzeigersinn“ (Counter-Clockwise)

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:
  - **Eingabe:** Anzahl an Knoten n, Anzahl an Ports von v, gesamte Historie von v (Protokoll aller bisher empfangen Nachrichten eines Knotens mit Zeitpunkt und Eingangsport)

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:
  - **Eingabe:** Anzahl an Knoten n, Anzahl an Ports von v, gesamte Historie von v (Protokoll aller bisher empfangen Nachrichten eines Knotens mit Zeitpunkt und Eingangsport)
  - **Ausgabe:** Zu sendende Nachrichten des Knotens an Ausgangsports und gegebenenfalls Entscheidung, ob v Leader oder Follower wird

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:
  - **Eingabe:** Anzahl an Knoten n, Anzahl an Ports von v, gesamte Historie von v (Protokoll aller bisher empfangen Nachrichten eines Knotens mit Zeitpunkt und Eingangsport)
  - **Ausgabe:** Zu sendende Nachrichten des Knotens an Ausgangsports und gegebenenfalls Entscheidung, ob v Leader oder Follower wird
- Funktion löst das Problem, wenn für jedes Netzwerk nach endlich vielen Anwendungen von f ein Knoten als Leader und alle anderen als Follower festgelegt wurden

Unmöglichkeits-Beweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Unmöglichkeits-Beweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn IH gilt:

- Wegen IH haben in jeder Runde alle Knoten die gleiche Historie

Unmöglichkeits-Beweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn IH gilt:

- Wegen IH haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich

Unmöglichkeits-Beweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn IH gilt:

- Wegen IH haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist in jeder Runde die Ausgabe der Funktion f für alle Knoten gleich

Unmöglichkeits-Beweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn IH gilt:

- Wegen IH haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist in jeder Runde die Ausgabe der Funktion f für alle Knoten gleich
- Somit: In jeder Runde macht f entweder alle Knoten zu Leadern, alle Knoten zu Followern oder trifft für keinen Knoten eine Entscheidung

Unmöglichkeits-Beweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn IH gilt:

- Wegen IH haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist in jeder Runde die Ausgabe der Funktion f für alle Knoten gleich
- Somit: In jeder Runde macht f entweder alle Knoten zu Leadern, alle Knoten zu Followern oder trifft für keinen Knoten eine Entscheidung
- f liefert also entweder ein falsches Ergebnis oder terminiert nicht

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da IH für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da IH für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da IH für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist auch die Ausgabe der Funktion f für alle Knoten gleich

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da IH für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist auch die Ausgabe der Funktion f für alle Knoten gleich
- Somit: Alle Knoten senden die jeweils gleiche Nachricht im und gegen den Uhrzeigersinn

Induktionsbeweis

Induktionshypothese

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da IH für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist auch die Ausgabe der Funktion f für alle Knoten gleich
- Somit: Alle Knoten senden die jeweils gleiche Nachricht im und gegen den Uhrzeigersinn
- Wegen Ring-Topologie: Alle empfangen jeweils gleiche Nachricht im und gegen den Uhrzeigersinn

IDs statt anonymer Knoten

Idee

Knoten mit kleinster ID wird zum Leader

IDs statt anonymer Knoten

Idee

Knoten mit kleinster ID wird zum Leader

Frage: Mit welchem Algorithmus kann dieser Knoten bestimmt werden?

IDs statt anonymer Knoten

Idee

Knoten mit kleinster ID wird zum Leader

Frage: Mit welchem Algorithmus kann dieser Knoten bestimmt werden?

Achtung!

Die kleinste vergebene ID ist nicht bekannt!

Problem wäre z. B. trivial, wenn es immer einen Knoten mit ID 0 gäbe

Clockwise Algorithmus (Synchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Clockwise Algorithmus (Synchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Jeder Knoten v führt folgenden Algorithmus aus:

Clockwise Algorithmus (Synchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1. v setzt $T_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
2. v sendet T_v an Nachbar im Uhrzeigersinn

Clockwise Algorithmus (Synchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1 v setzt $T_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)

2 v sendet T_v an Nachbar im Uhrzeigersinn

Runde $r \geq 2$:

1 **receive** M **at** v**:**

Clockwise Algorithmus (Synchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1 v setzt $T_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
2 v sendet T_v an Nachbar im Uhrzeigersinn

Runde $r \geq 2$:

1 **receive** M **at** v**:**
2 **if** $M < T_v$ **then**
3 v setzt $T_v := M$
4 v wird zum Follower (sofern nicht bereits vorher geschehen)
5 v sendet T_v an Nachbar im Uhrzeigersinn

Clockwise Algorithmus (Synchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Jeder Knoten v führt folgenden Algorithmus aus:

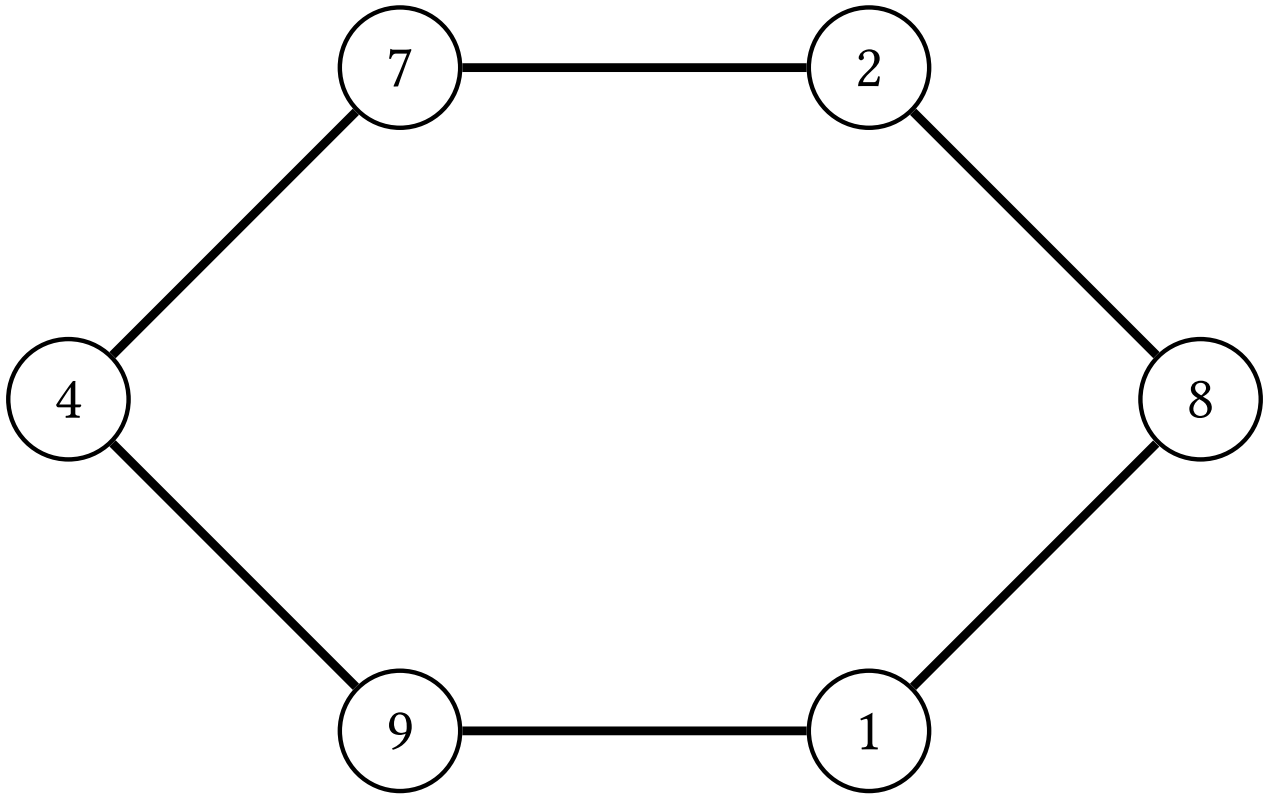
Runde 1:

1 v setzt $T_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
2 v sendet T_v an Nachbar im Uhrzeigersinn

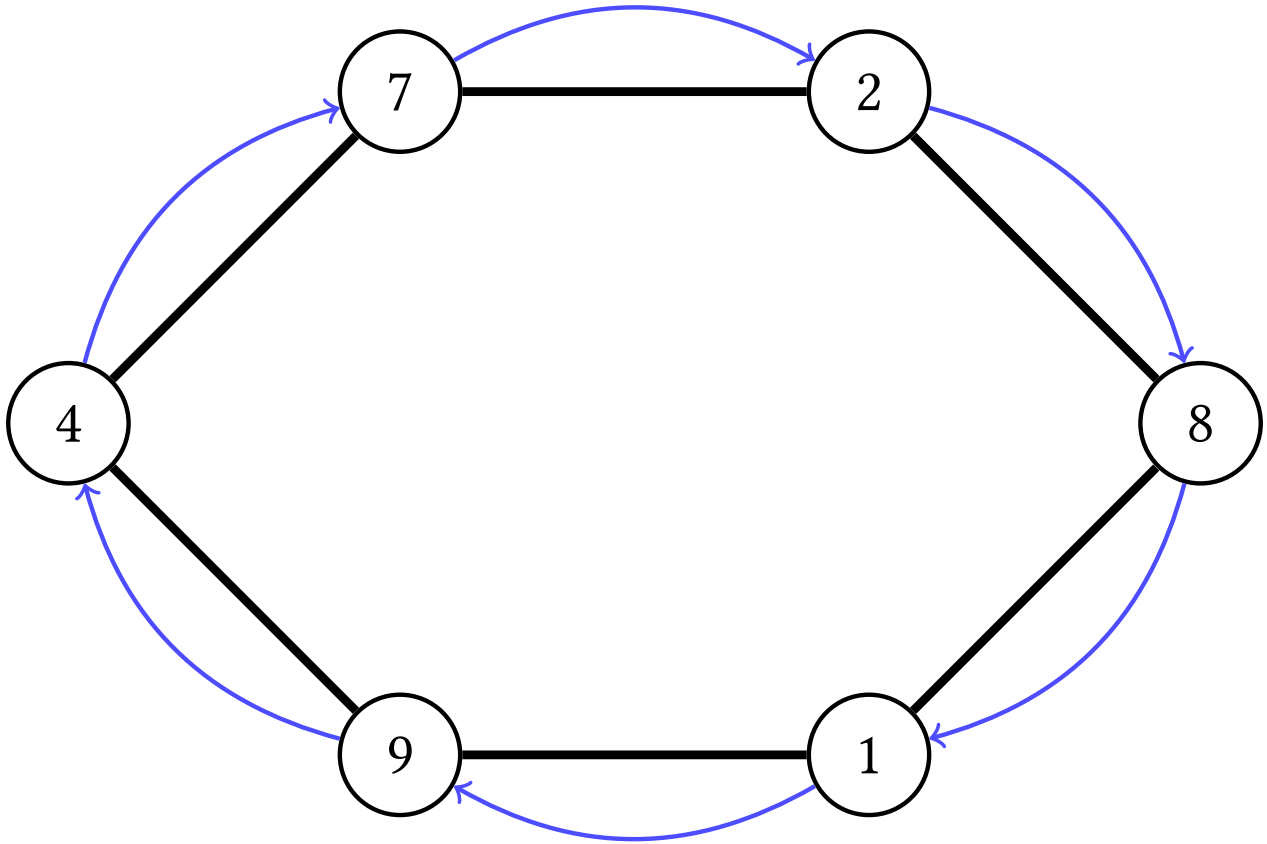
Runde $r \geq 2$:

1 **receive** M **at** v:
2 **if** $M < T_v$ **then**
3 v setzt $T_v := M$
4 v wird zum Follower (sofern nicht bereits vorher geschehen)
5 v sendet T_v an Nachbar im Uhrzeigersinn
6 **if** $M = \mathrm{ID}(v)$ **then**
7 v wird zum Leader

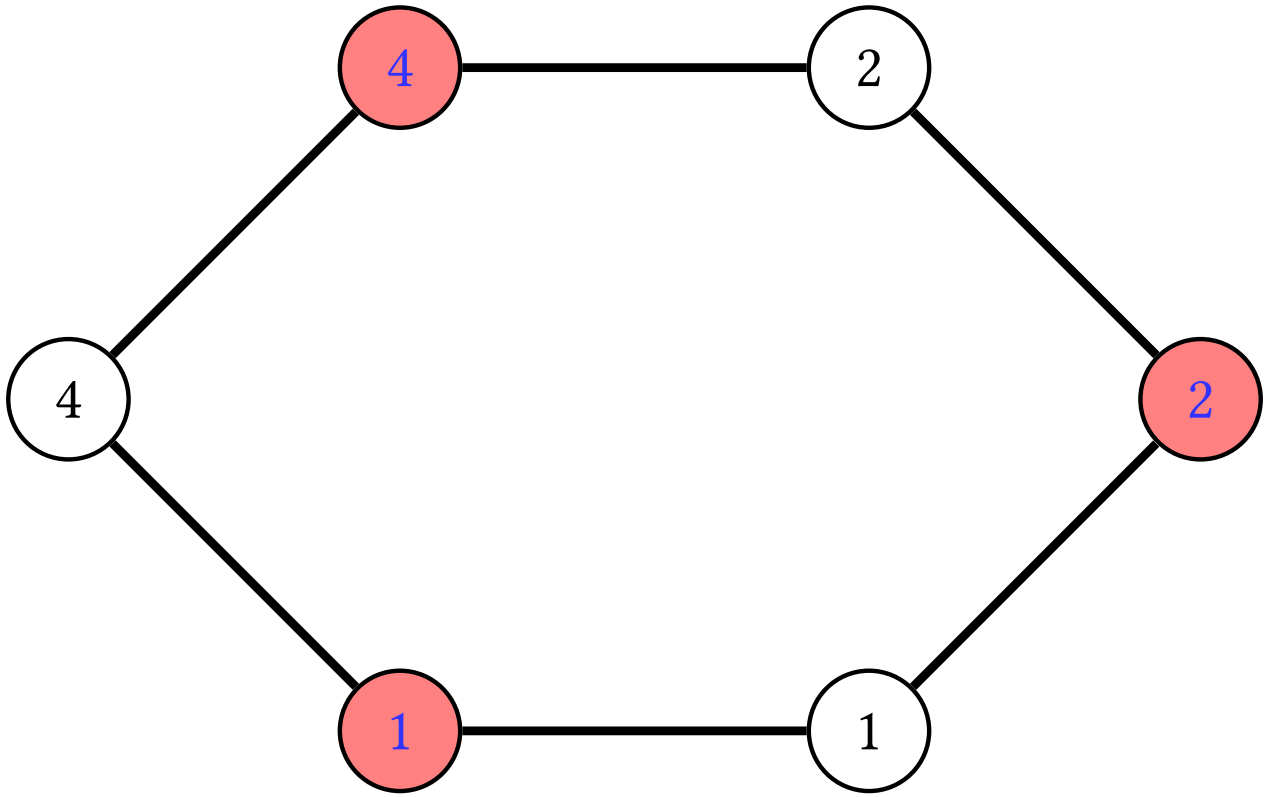
Clockwise Algorithmus (Synchron): Beispiel



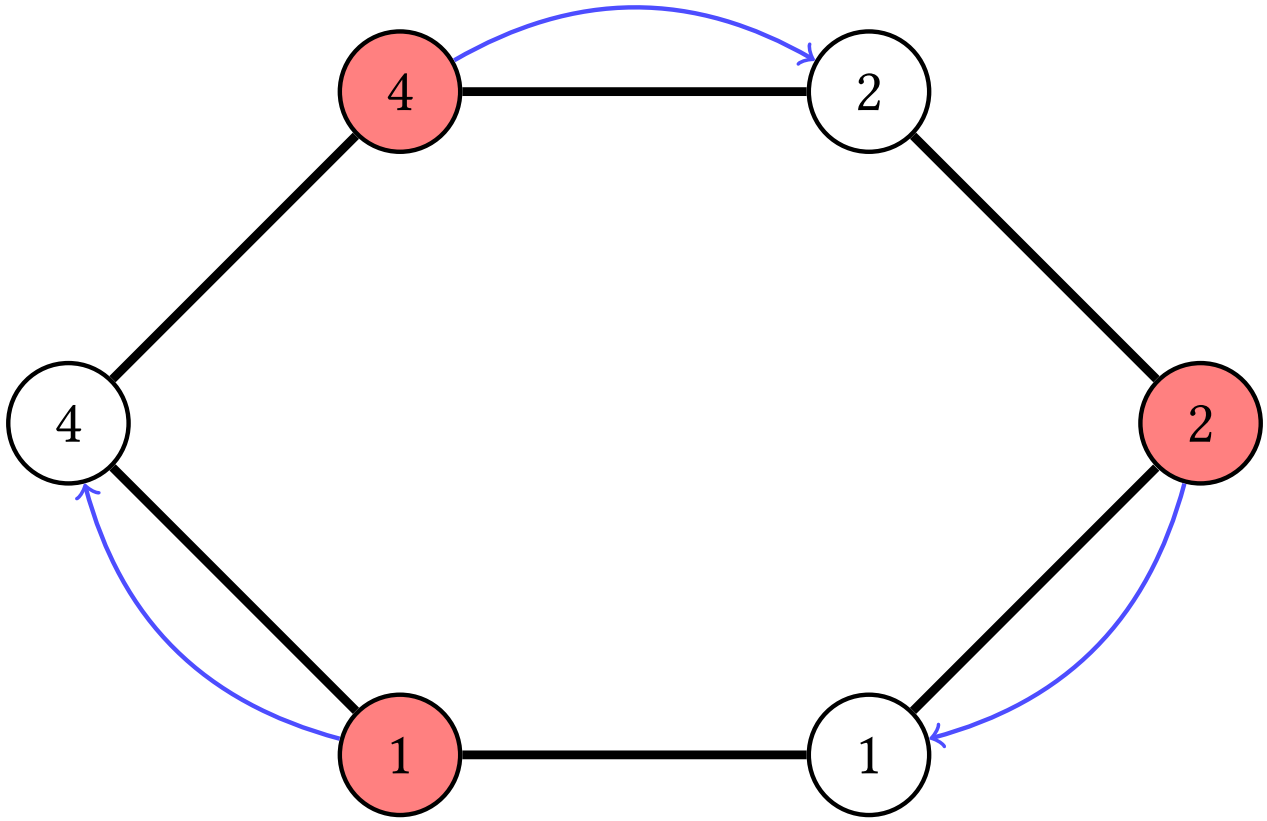
Clockwise Algorithmus (Synchron): Beispiel



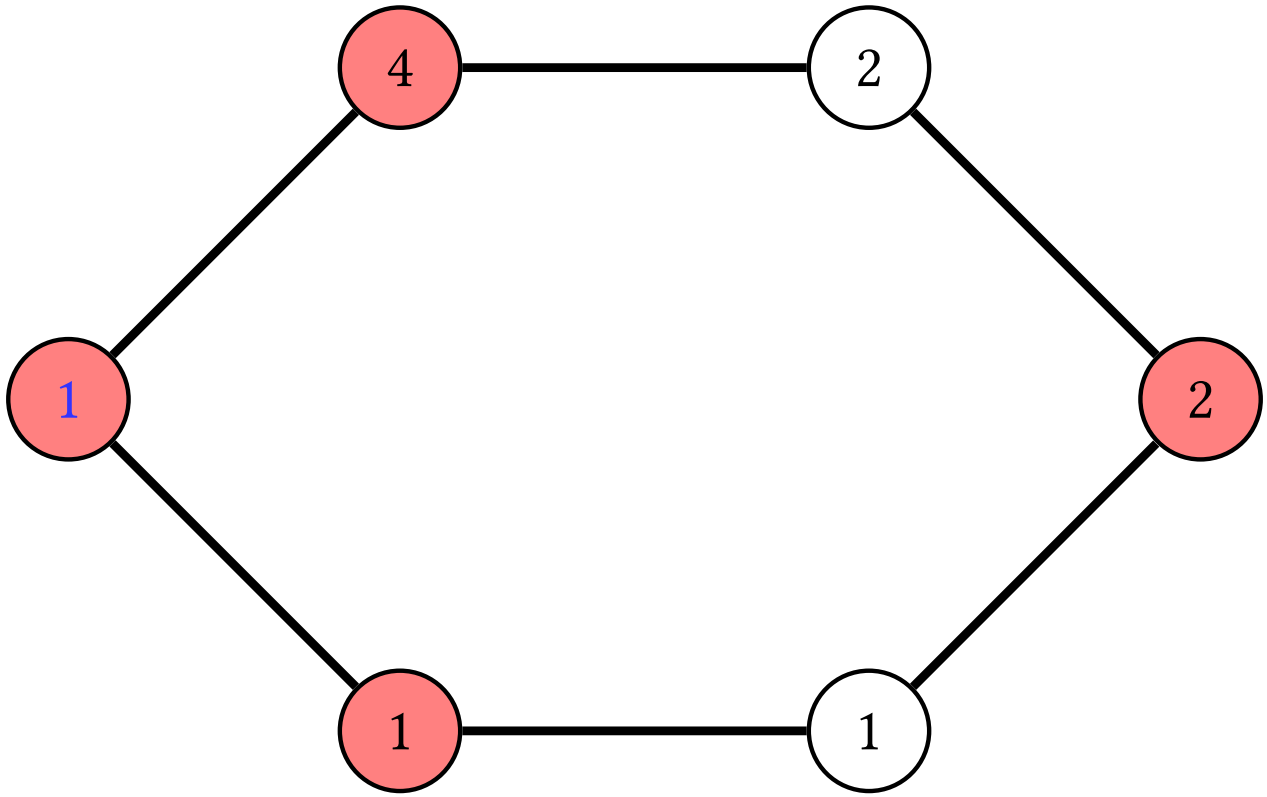
Clockwise Algorithmus (Synchron): Beispiel



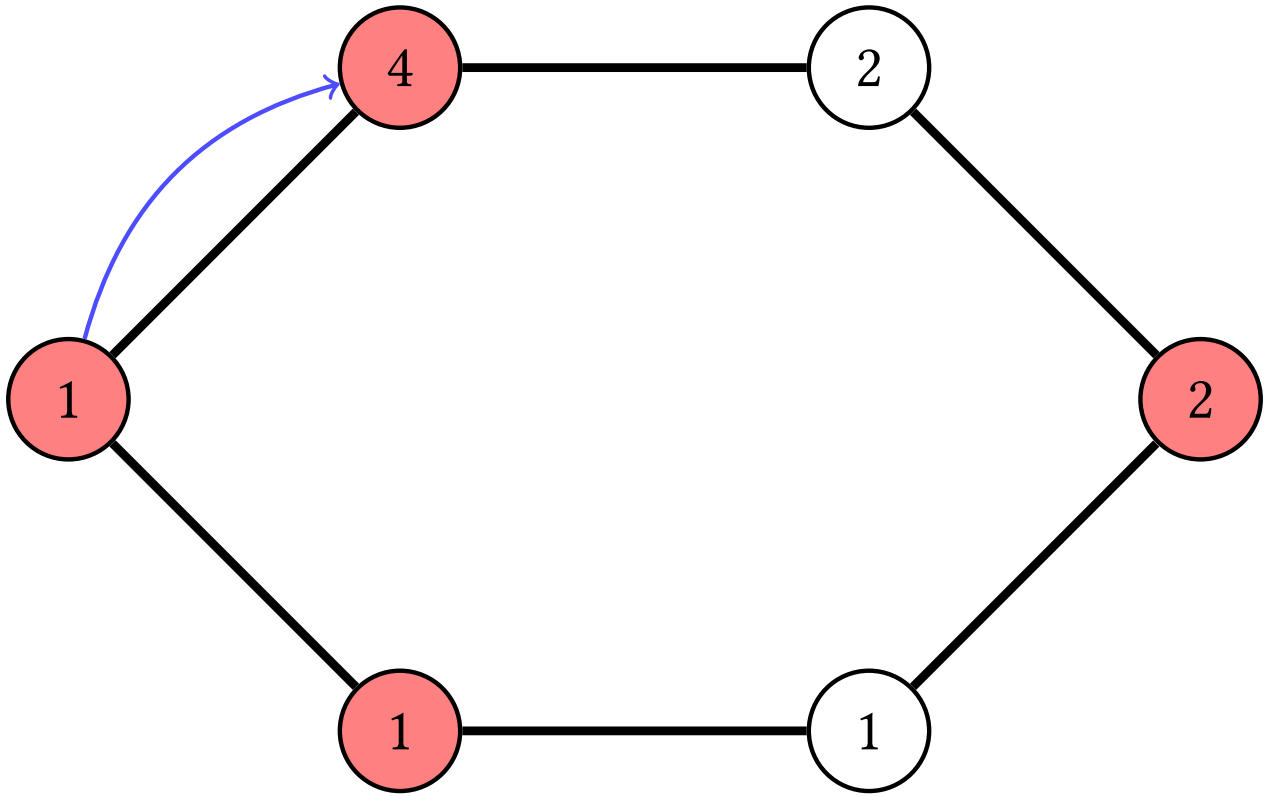
Clockwise Algorithmus (Synchron): Beispiel



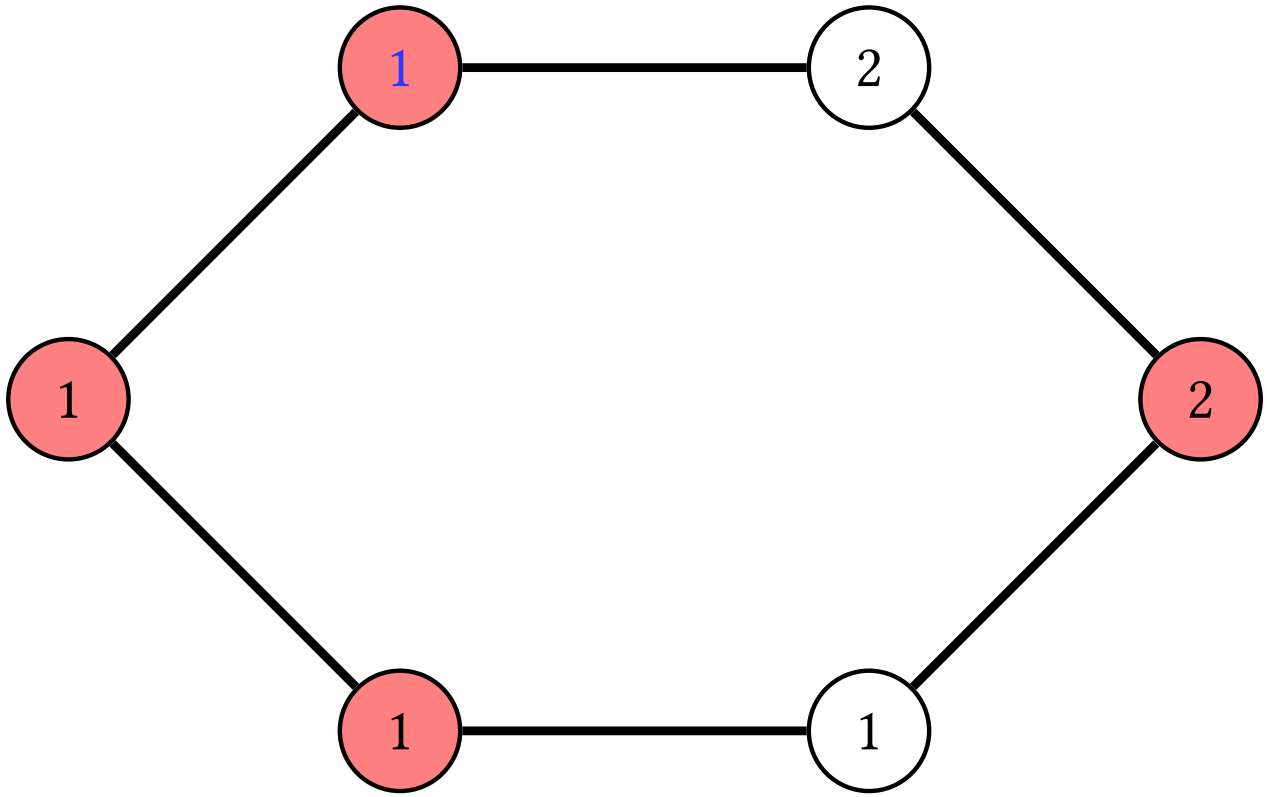
Clockwise Algorithmus (Synchron): Beispiel



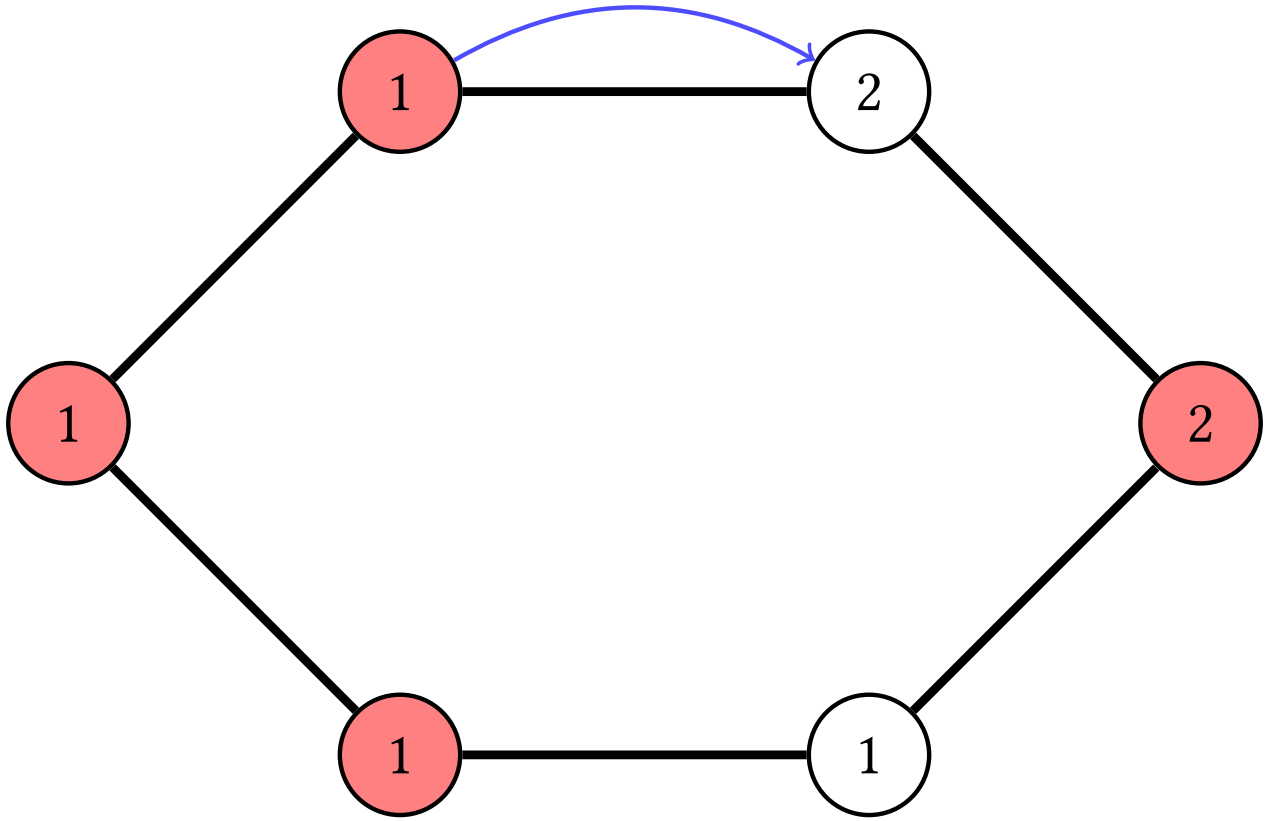
Clockwise Algorithmus (Synchron): Beispiel



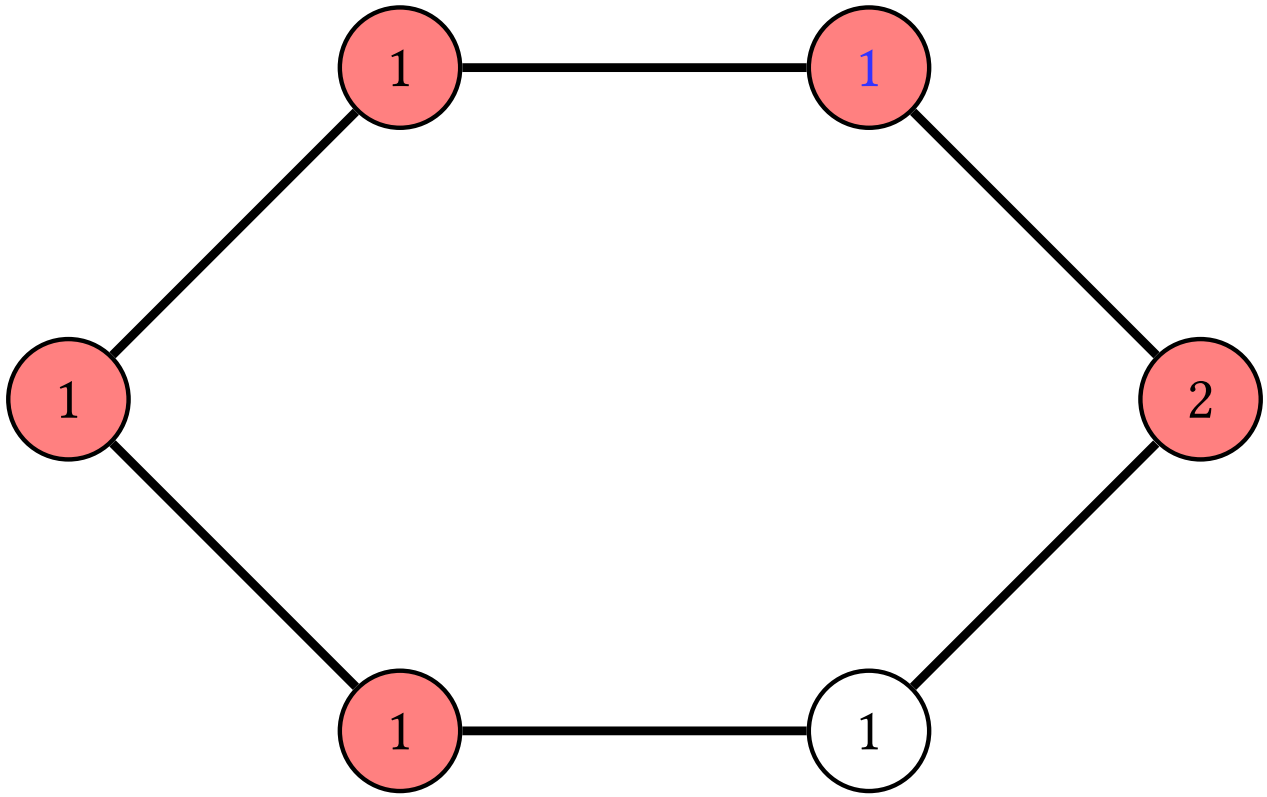
Clockwise Algorithmus (Synchron): Beispiel



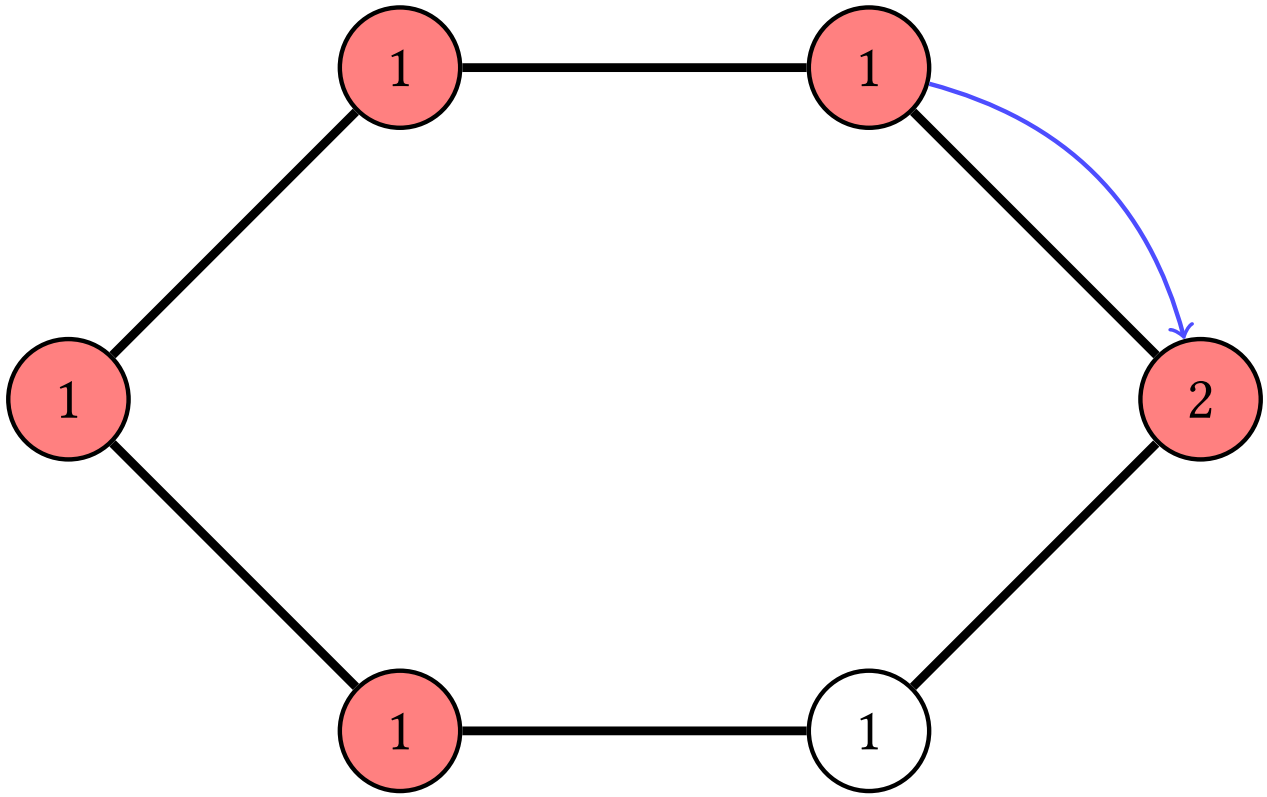
Clockwise Algorithmus (Synchron): Beispiel



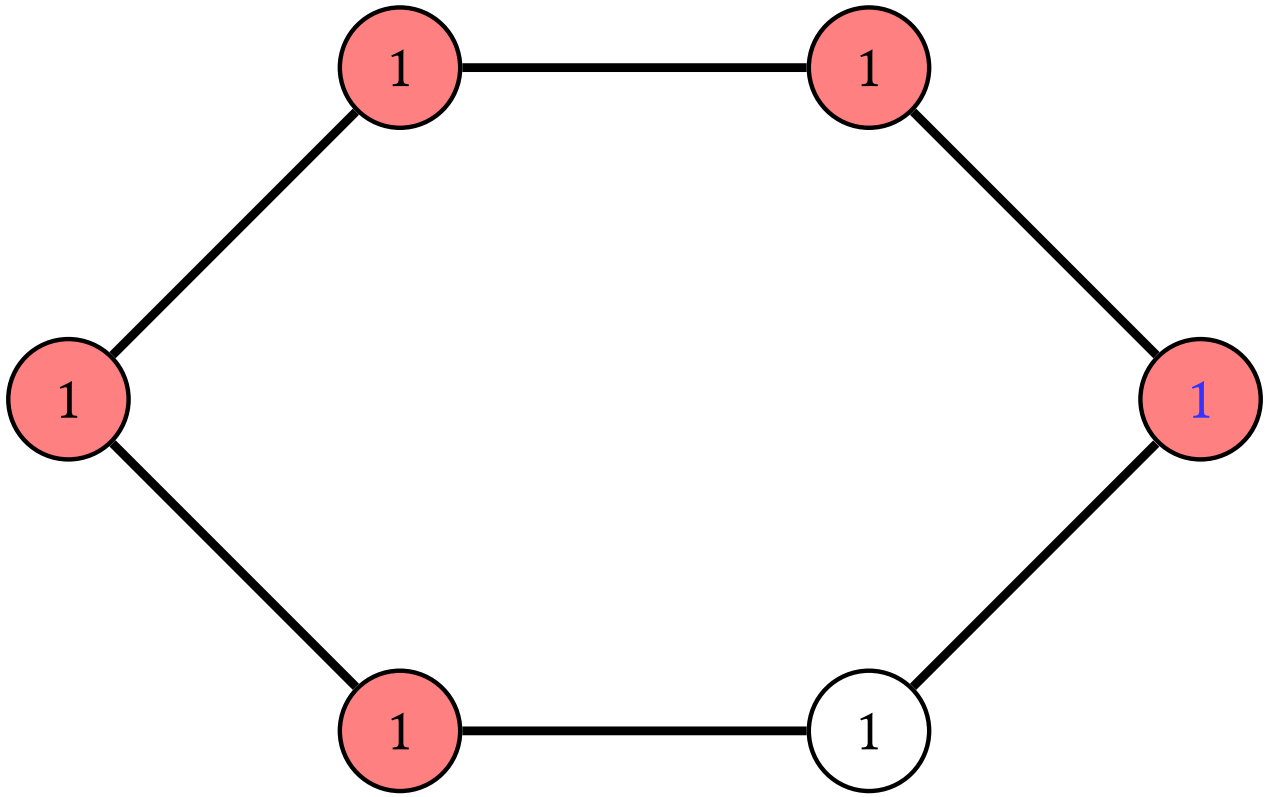
Clockwise Algorithmus (Synchron): Beispiel



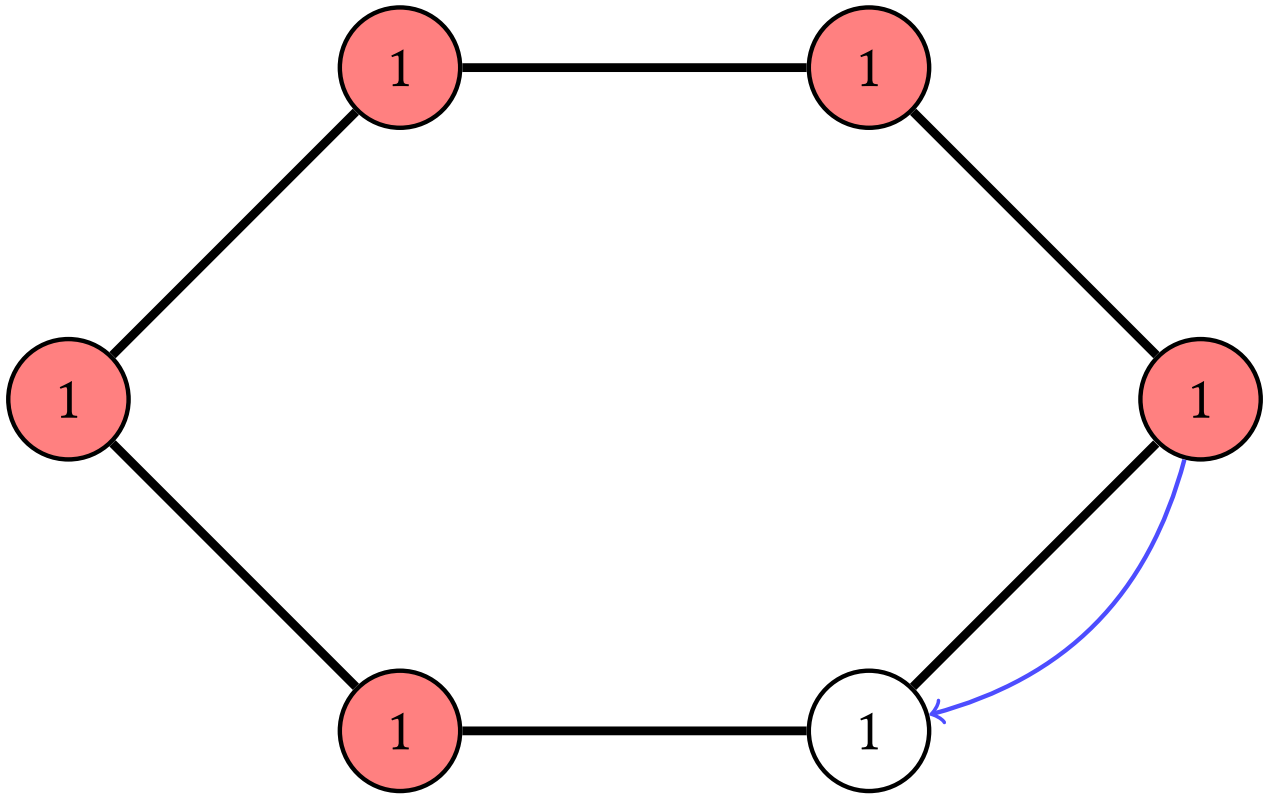
Clockwise Algorithmus (Synchron): Beispiel



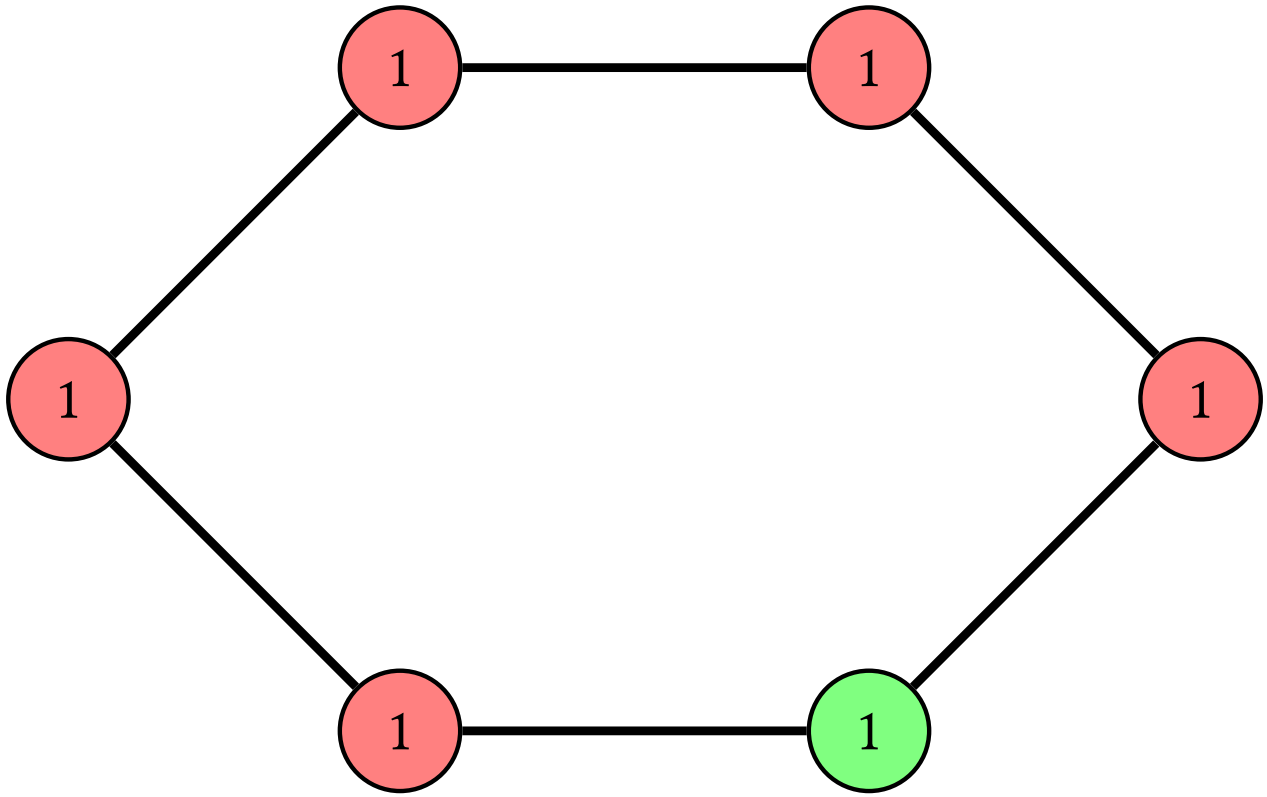
Clockwise Algorithmus (Synchron): Beispiel



Clockwise Algorithmus (Synchron): Beispiel



Clockwise Algorithmus (Synchron): Beispiel



Laufzeitanalyse Clockwise Algorithmus (Synchron)

Theorem

Nach höchstens $n + 1$ Runden bestimmt der synchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Laufzeitanalyse Clockwise Algorithmus (Synchron)

Theorem

Nach höchstens $n + 1$ Runden bestimmt der synchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq T_v$ für jeden Knoten v

Laufzeitanalyse Clockwise Algorithmus (Synchron)

Theorem

Nach höchstens $n + 1$ Runden bestimmt der synchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq T_v$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.

Laufzeitanalyse Clockwise Algorithmus (Synchron)

Theorem

Nach höchstens $n + 1$ Runden bestimmt der synchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq T_v$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.
- **Induktionshypothese:** In Runde $i + 1$ empfängt v_i die Nachricht $\mathrm{ID}(z)$ (für jedes $i \leq n$)

Laufzeitanalyse Clockwise Algorithmus (Synchron)

Theorem

Nach höchstens $n + 1$ Runden bestimmt der synchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq T_v$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.
- **Induktionshypothese:** In Runde $i + 1$ empfängt v_i die Nachricht $\mathrm{ID}(z)$ (für jedes $i \leq n$)
- **Somit:**
  - In Runde $n + 1$ empfängt z seine eigene ID und wird zum Leader

Laufzeitanalyse Clockwise Algorithmus (Synchron)

Theorem

Nach höchstens $n + 1$ Runden bestimmt der synchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq T_v$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.
- **Induktionshypothese:** In Runde $i + 1$ empfängt v_i die Nachricht $\mathrm{ID}(z)$ (für jedes $i \leq n$)
- **Somit:**
  - In Runde $n + 1$ empfängt z seine eigene ID und wird zum Leader
  - Sei $v \neq z$. Dann ist $v = v_i$ für ein $i \leq n - 1$. Spätestens in Runde $i + 1 \leq n$ empfängt v_i die Nachricht $\mathrm{ID}(z)$ und wird zum Follower.

Nachrichtenkomplexität Clockwise Algorithmus

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Nachrichtenkomplexität Clockwise Algorithmus

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem T_v initialisiert oder verringert wurde.

Nachrichtenkomplexität Clockwise Algorithmus

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem T_v initialisiert oder verringert wurde.
- Jeder von T_v angenommene Wert entspricht einer ID im Netzwerk.

Nachrichtenkomplexität Clockwise Algorithmus

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem T_v initialisiert oder verringert wurde.
- Jeder von T_v angenommene Wert entspricht einer ID im Netzwerk.
- Daher kann T_v während des Algorithmus höchstens n verschiedene Werte annehmen.

Nachrichtenkomplexität Clockwise Algorithmus

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem T_v initialisiert oder verringert wurde.
- Jeder von T_v angenommene Wert entspricht einer ID im Netzwerk.
- Daher kann T_v während des Algorithmus höchstens n verschiedene Werte annehmen.
- Somit sendet jeder Knoten höchstens n Nachrichten.

Clockwise Algorithmus (Asynchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Clockwise Algorithmus (Asynchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

Clockwise Algorithmus (Asynchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

Jeder Knoten v führt folgenden Algorithmus aus:

1 **initialize** v**:**
2 $\quad$ v setzt $T_v := \mathrm{ID}(v)$ (lokale Variable für größte bisher gesehene ID)
3 $\quad$ v sendet T_v an Nachbar im Uhrzeigersinn

Clockwise Algorithmus (Asynchron) [LeLann '77, Chang/Roberts '79]

Annahme: Jeder Knoten v hat eindeutigen Identifier $\mathrm{ID}(v)$

Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

Jeder Knoten v führt folgenden Algorithmus aus:

```
1  initialize v:
2      v setzt T_v := ID(v) (lokale Variable für größte bisher gesehene ID)
3      v sendet T_v an Nachbar im Uhrzeigersinn
4  receive M at v:
5      if T_v uninitialisiert then T_v := ID(v)
6      if M < T_v then
7          v setzt T_v := M
8          v wird zum Follower (sofern nicht bereits vorher geschehen)
9          v sendet T_v an Nachbar im Uhrzeigersinn
10     if M = ID(v) then
11         v wird zum Leader zu sein
```

Analyse Clockwise Algorithmus (Asynchron)

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Analyse Clockwise Algorithmus (Asynchron)

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.

Analyse Clockwise Algorithmus (Asynchron)

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.
- Dann vergehen höchstens $n - 1$ Zeiteinheit bis Knoten z mit kleinster ID eine Nachricht empfängt und das erste Mal aktiv wird.

Analyse Clockwise Algorithmus (Asynchron)

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.
- Dann vergehen höchstens $n - 1$ Zeiteinheit bis Knoten z mit kleinster ID eine Nachricht empfängt und das erste Mal aktiv wird.
- Anschließend: Nach höchstens n weiteren Zeiteinheiten hat jeder Knoten $\mathrm{ID}(z)$ empfangen und sich entschieden.

Analyse Clockwise Algorithmus (Asynchron)

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.
- Dann vergehen höchstens $n - 1$ Zeiteinheit bis Knoten z mit kleinster ID eine Nachricht empfängt und das erste Mal aktiv wird.
- Anschließend: Nach höchstens n weiteren Zeiteinheiten hat jeder Knoten $\mathrm{ID}(z)$ empfangen und sich entschieden.

Nachrichtenkomplexität: $O(n^2)$
Gleiches Argument wie für synchronen Algorithmus

Verbesserung der Nachrichtenkomplexität

Frage

Kann ein Leader mit signifikant weniger als $\Theta(n^2)$ Nachrichten bestimmt werden?

Verbesserung der Nachrichtenkomplexität

Frage

Kann ein Leader mit signifikant weniger als $\Theta(n^2)$ Nachrichten bestimmt werden?

„Verantwortlich ist man nicht nur, für das, was man tut, sondern auch für das, was man nicht tut.“

Verbesserung der Nachrichtenkomplexität

Frage

Kann ein Leader mit signifikant weniger als $\Theta(n^2)$ Nachrichten bestimmt werden?

„Verantwortlich ist man nicht nur, für das, was man tut, sondern auch für das, was man nicht tut.“

Idee: *Keine* Nachricht zu senden darf als Zustimmung interpretiert werden

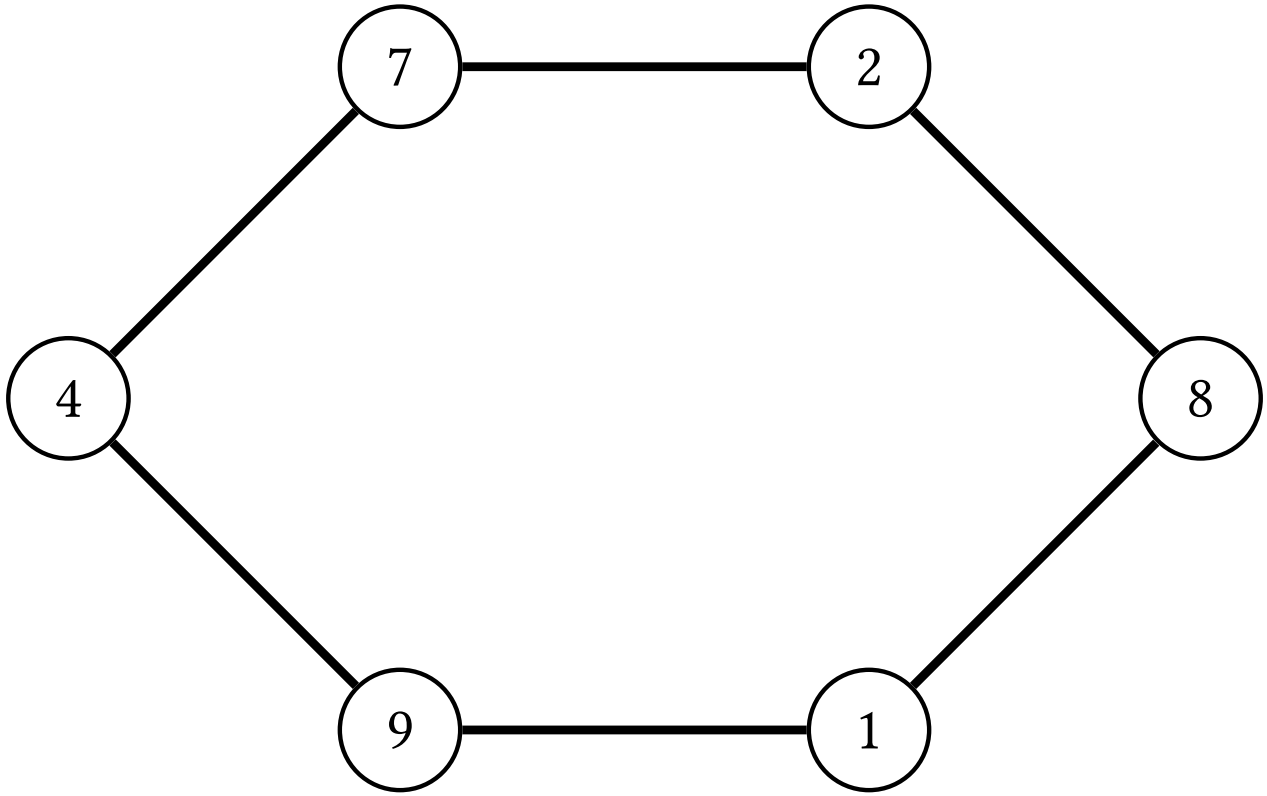
Nachrichteneffiziente Algorithmus (Synchron)

Jeder Knoten v führt folgenden Algorithmus in jeder Runde aus:

1 **if** *Leader-Nachricht empfangen undn noch kein Leader* **then**
2 | Werde zum Follower
3 | Leite Leader-Nachricht im Uhrzeigersinn
4 **if** $\#Runden = \mathrm{ID}(v) * n + 1$ *und* v *noch kein Follower* **then**
5 | Werde zum Leader
6 | Sende Leader-Nachricht an Nachbar im Uhrzeigersinn

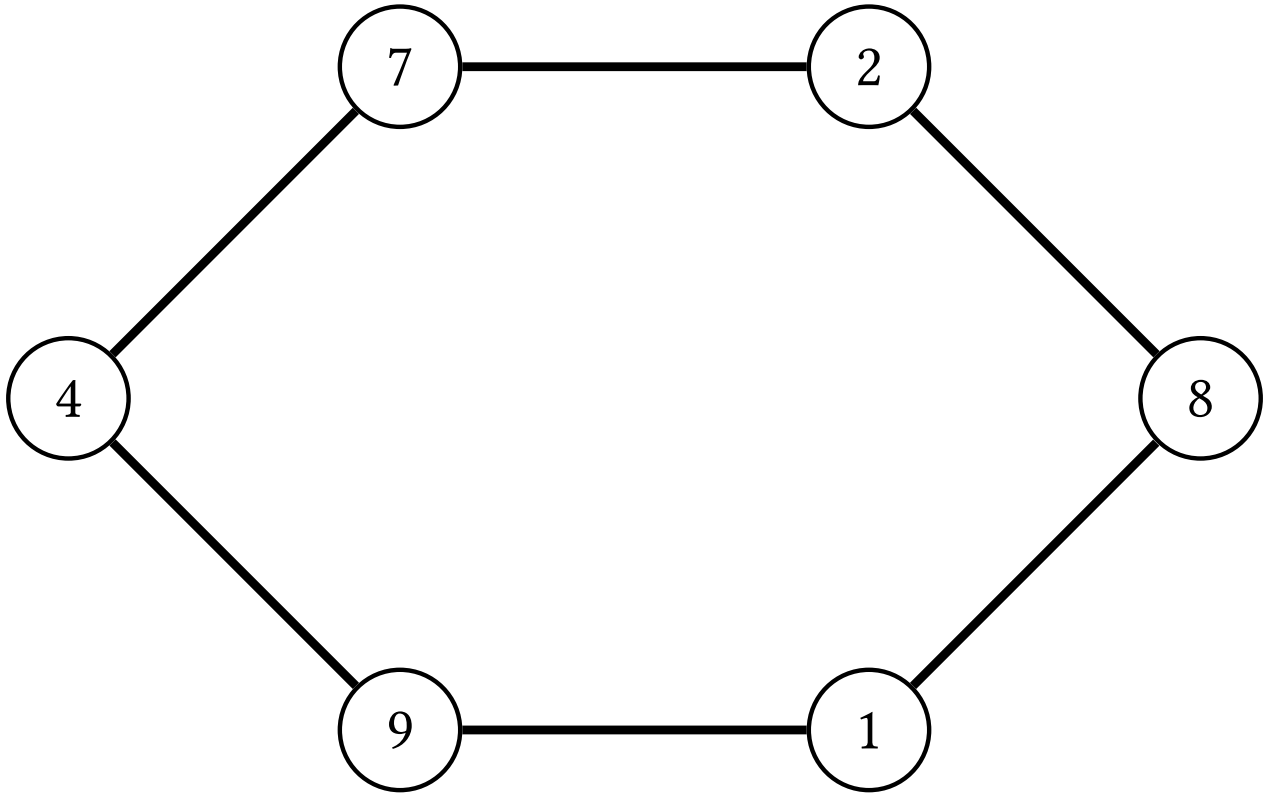
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 1:



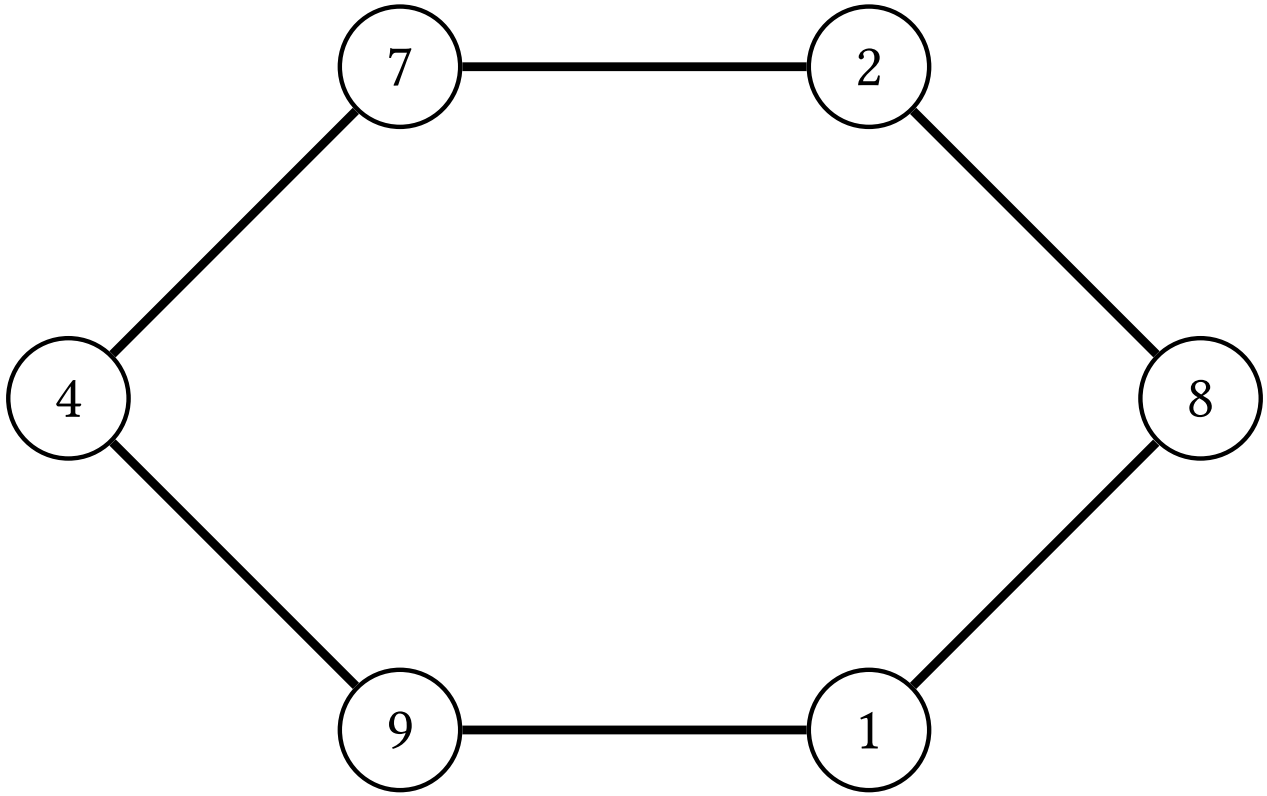
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 2:



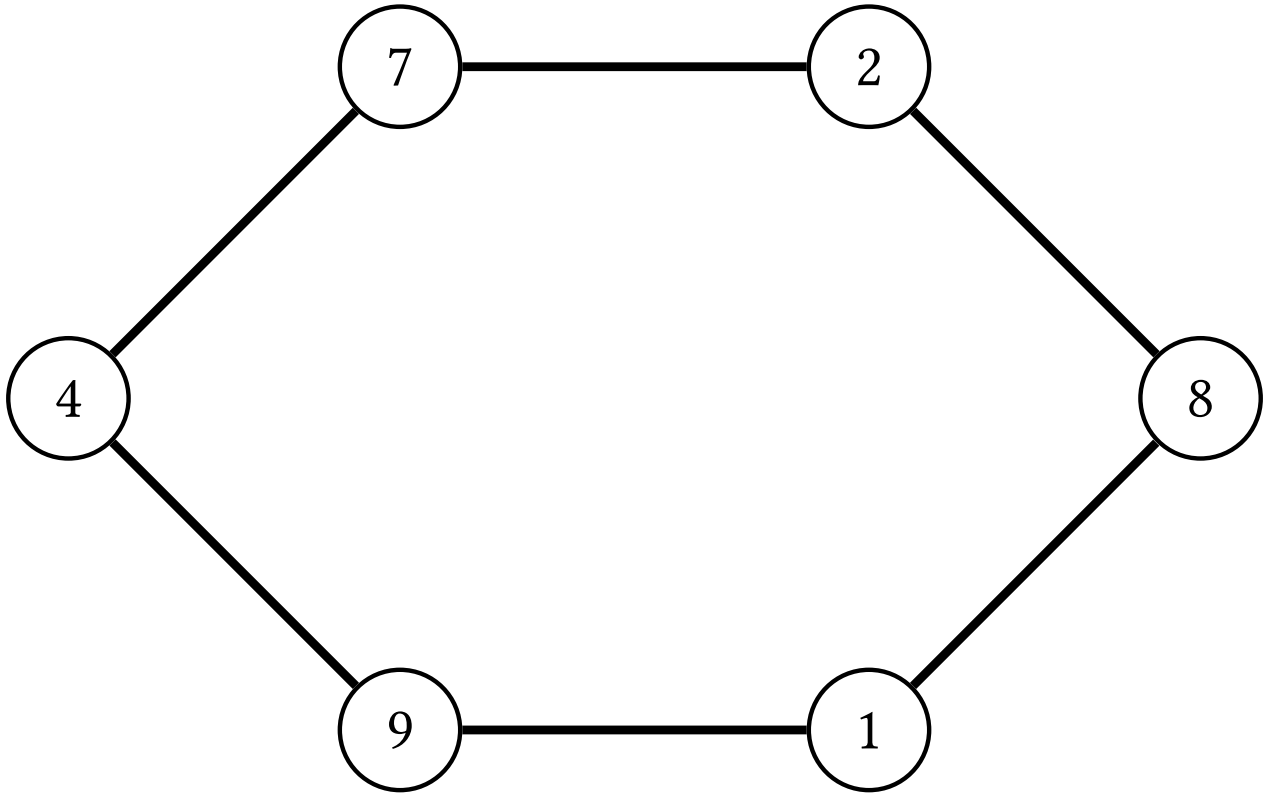
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 3:



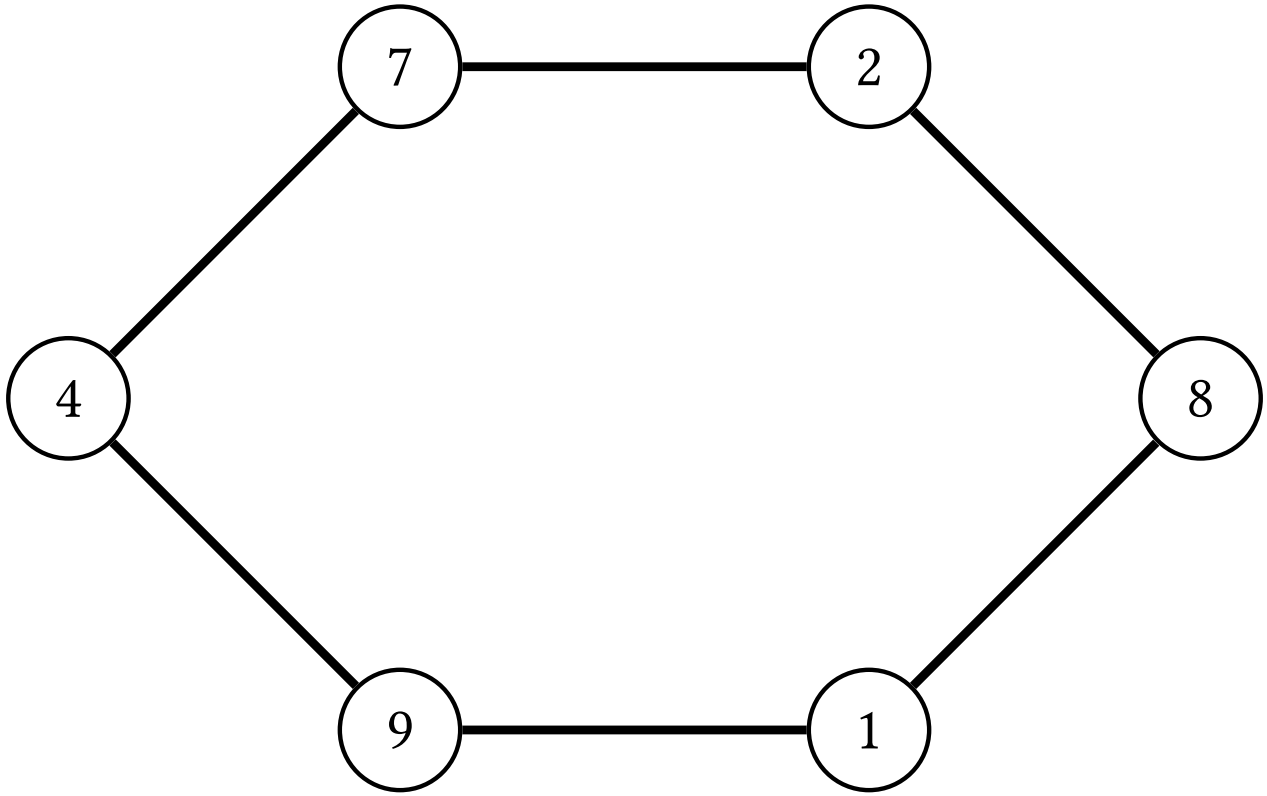
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 4:



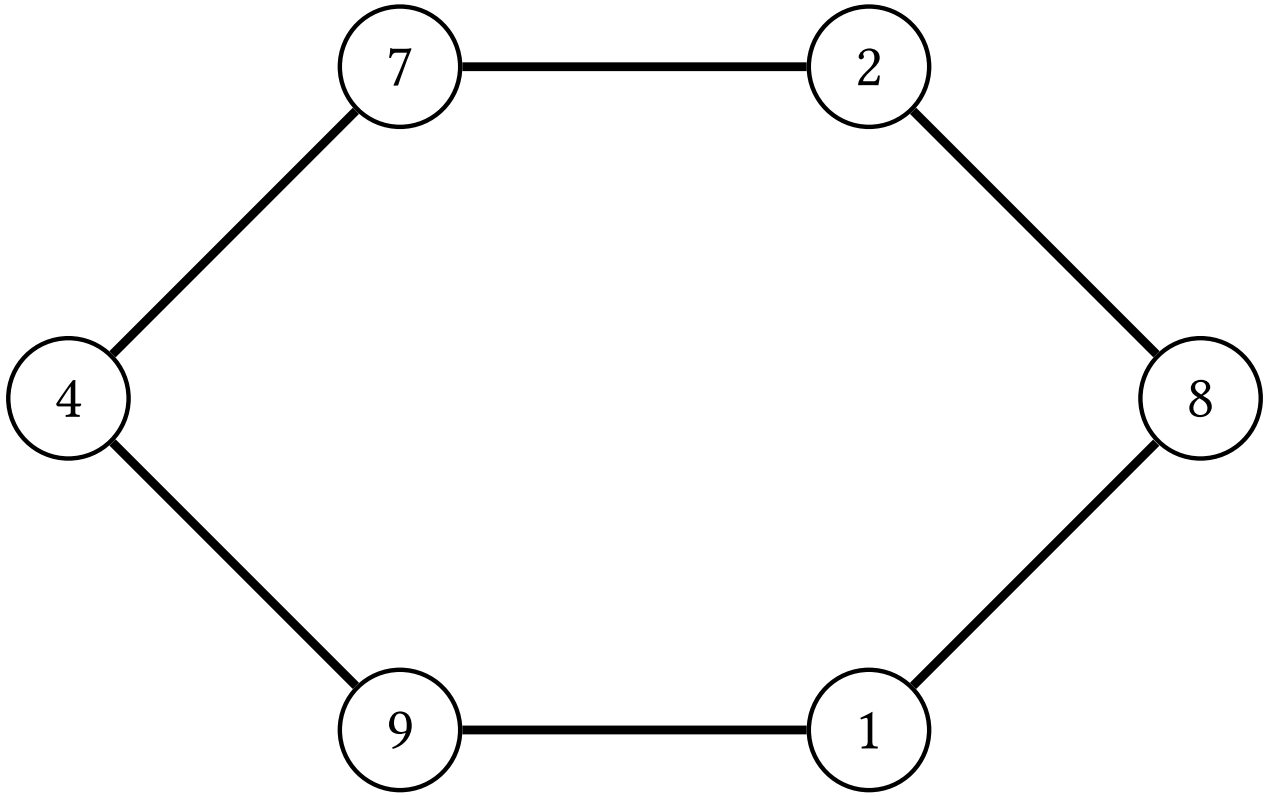
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 5:



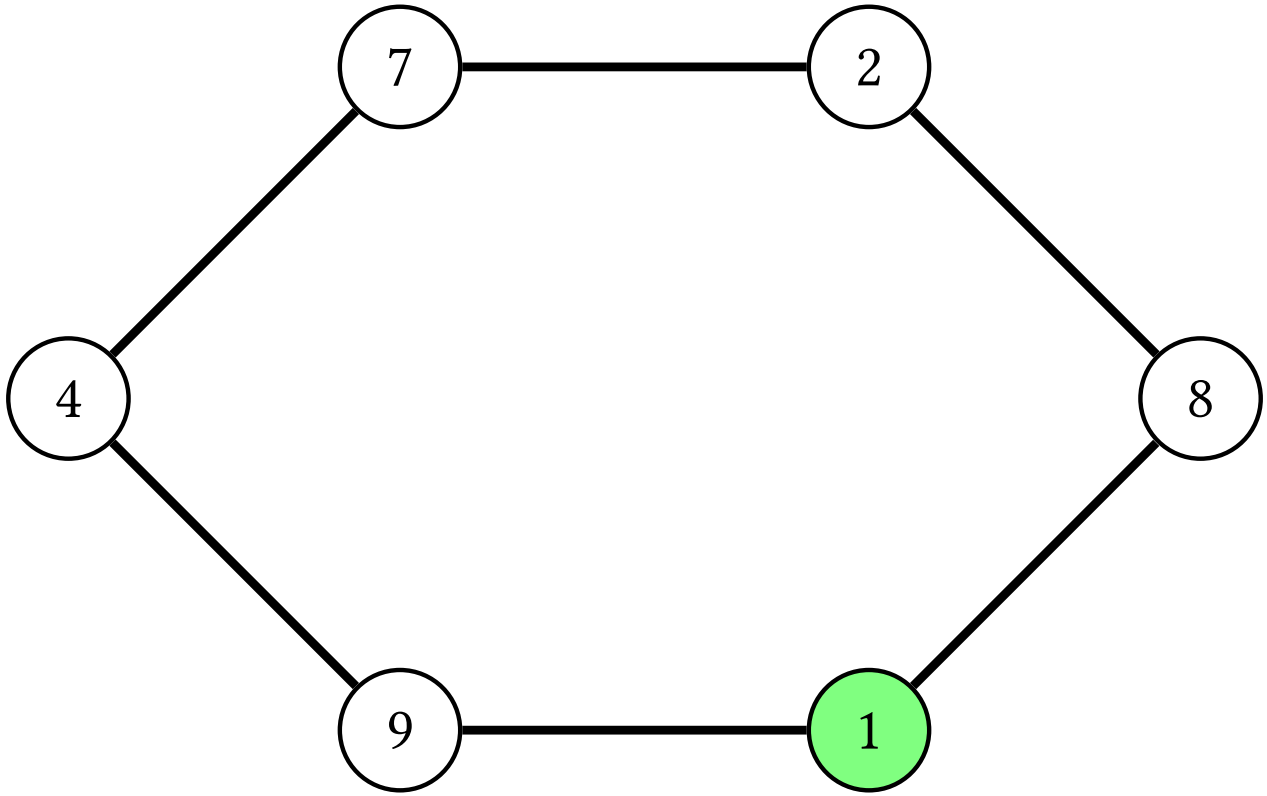
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 6:



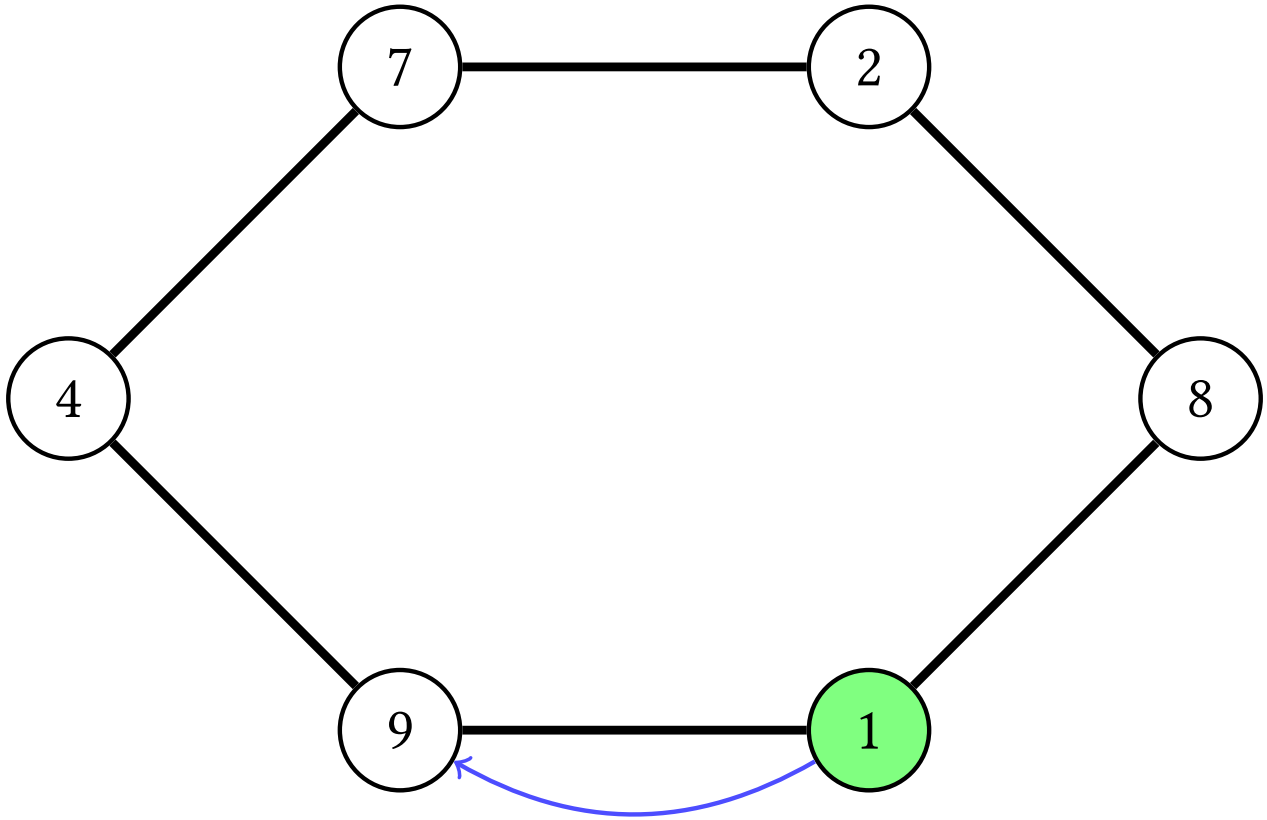
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 7:



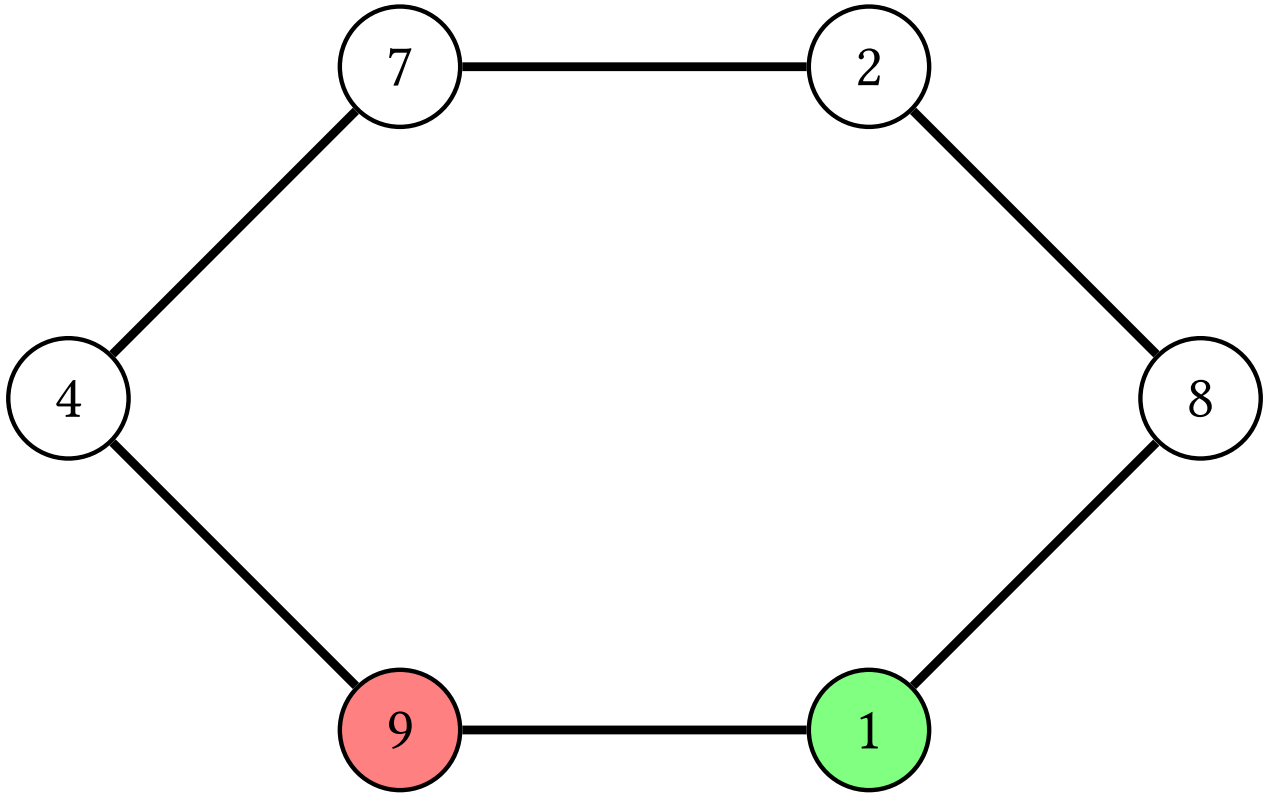
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 7:



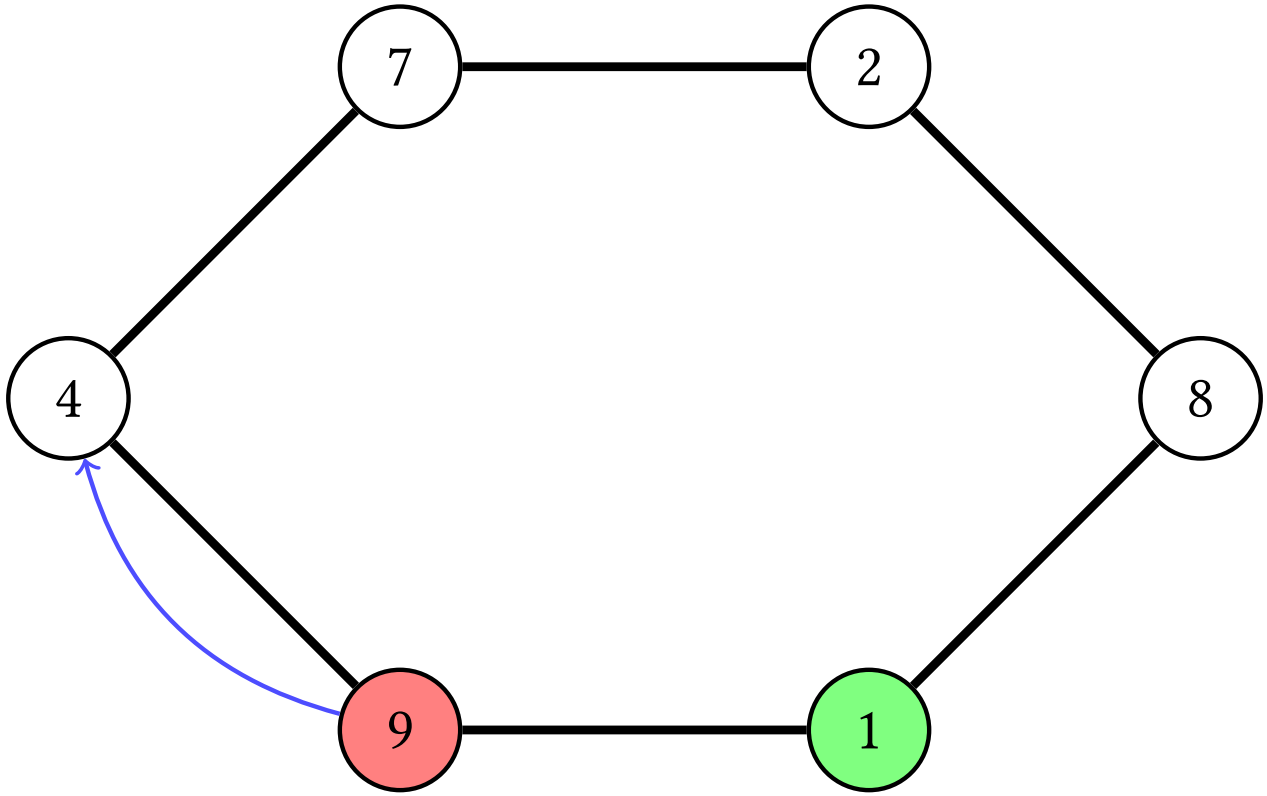
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 8:



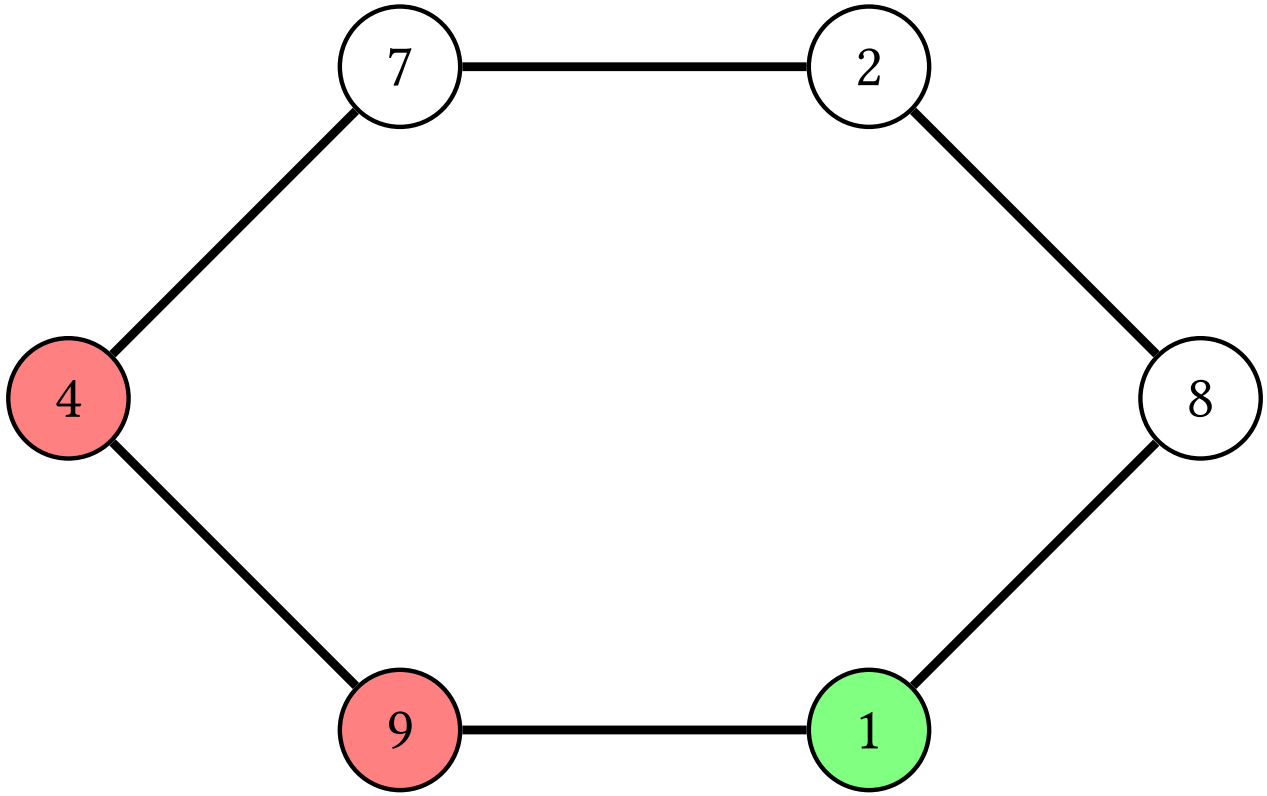
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 8:



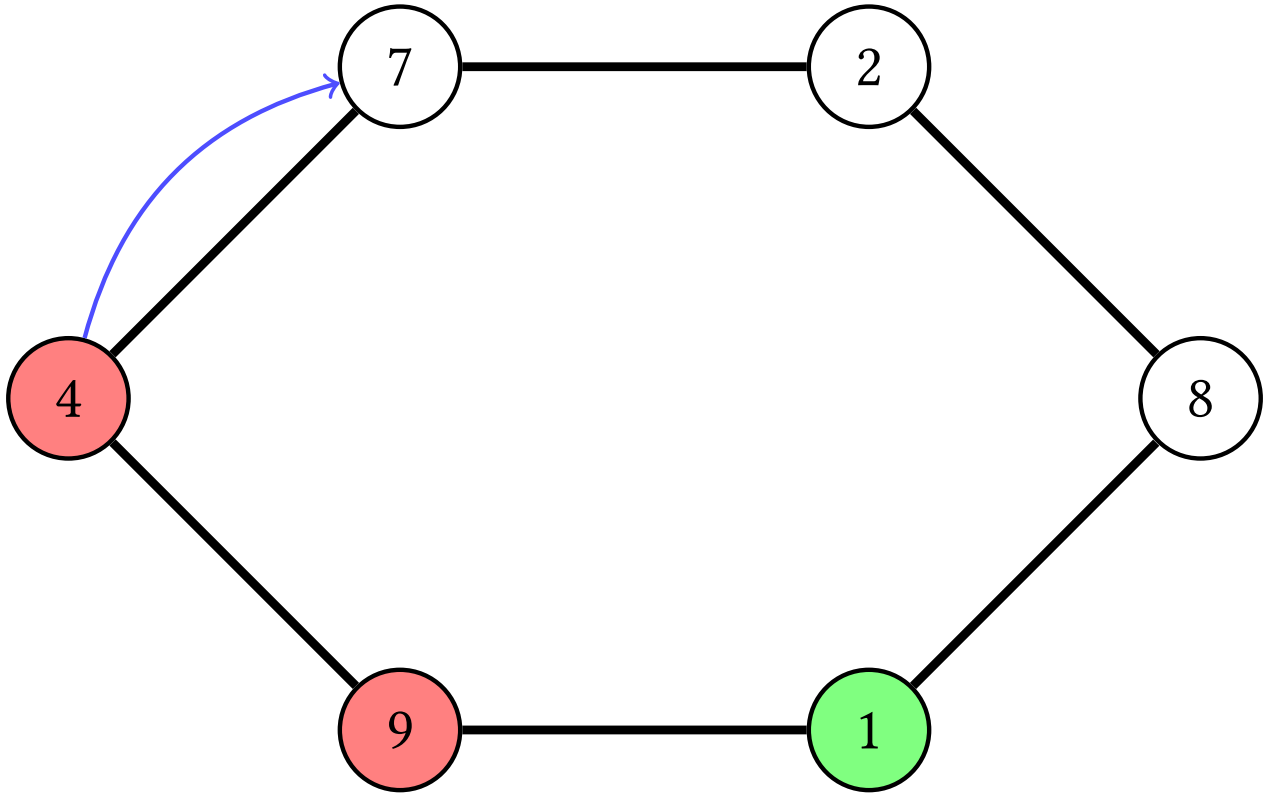
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 9:



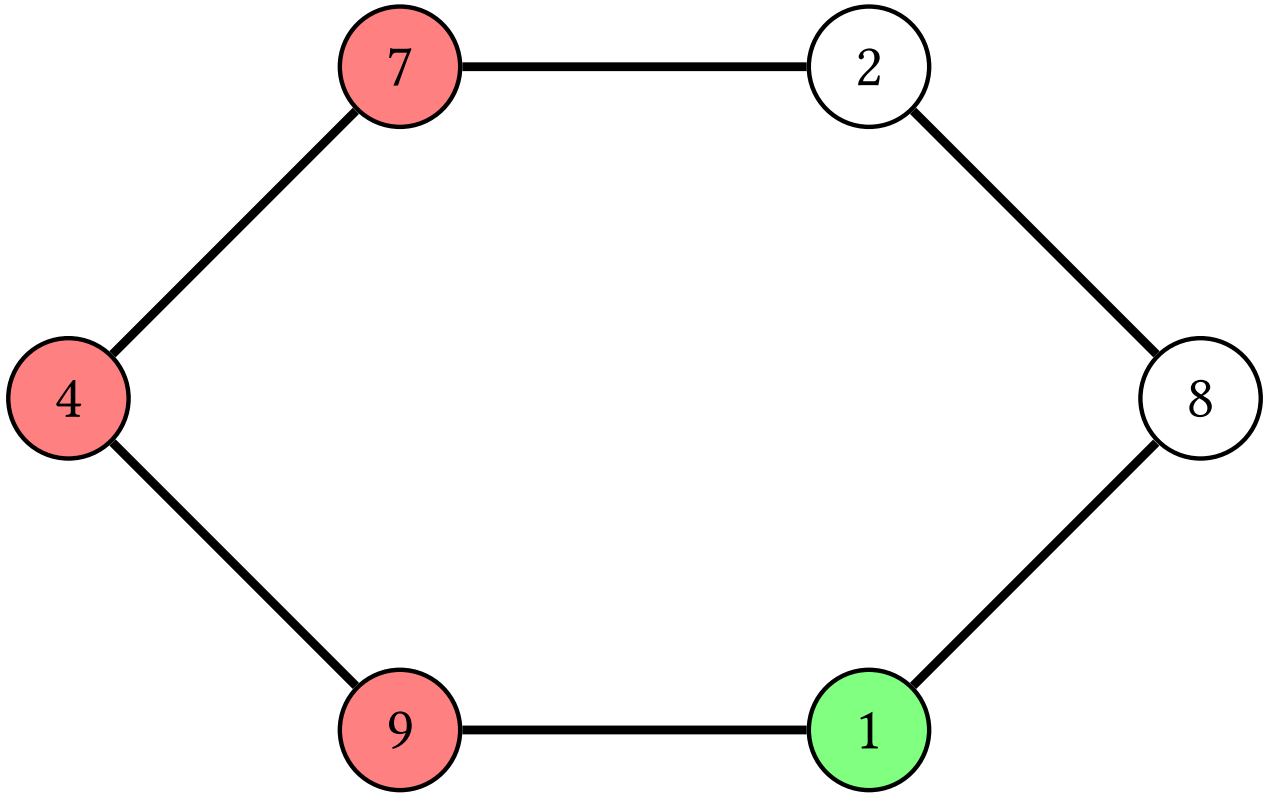
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 9:



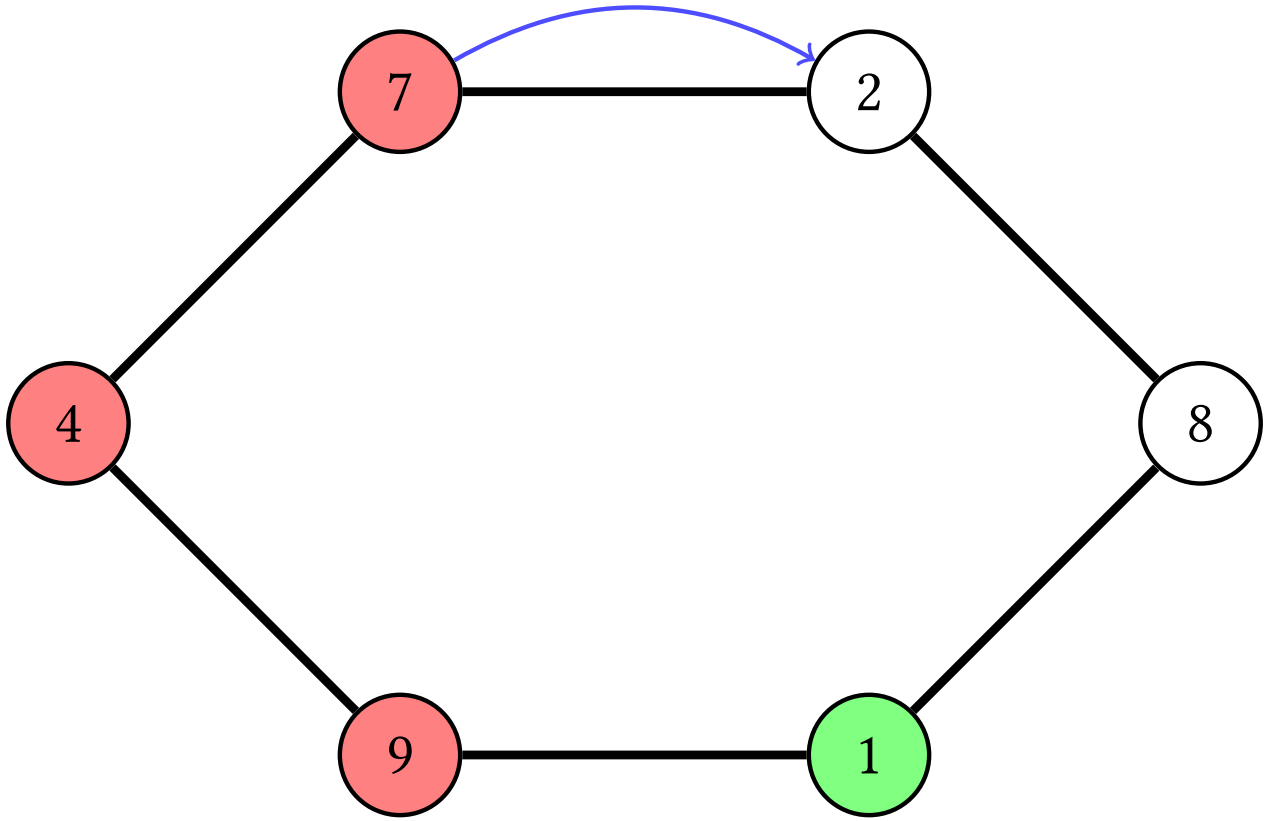
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 10:



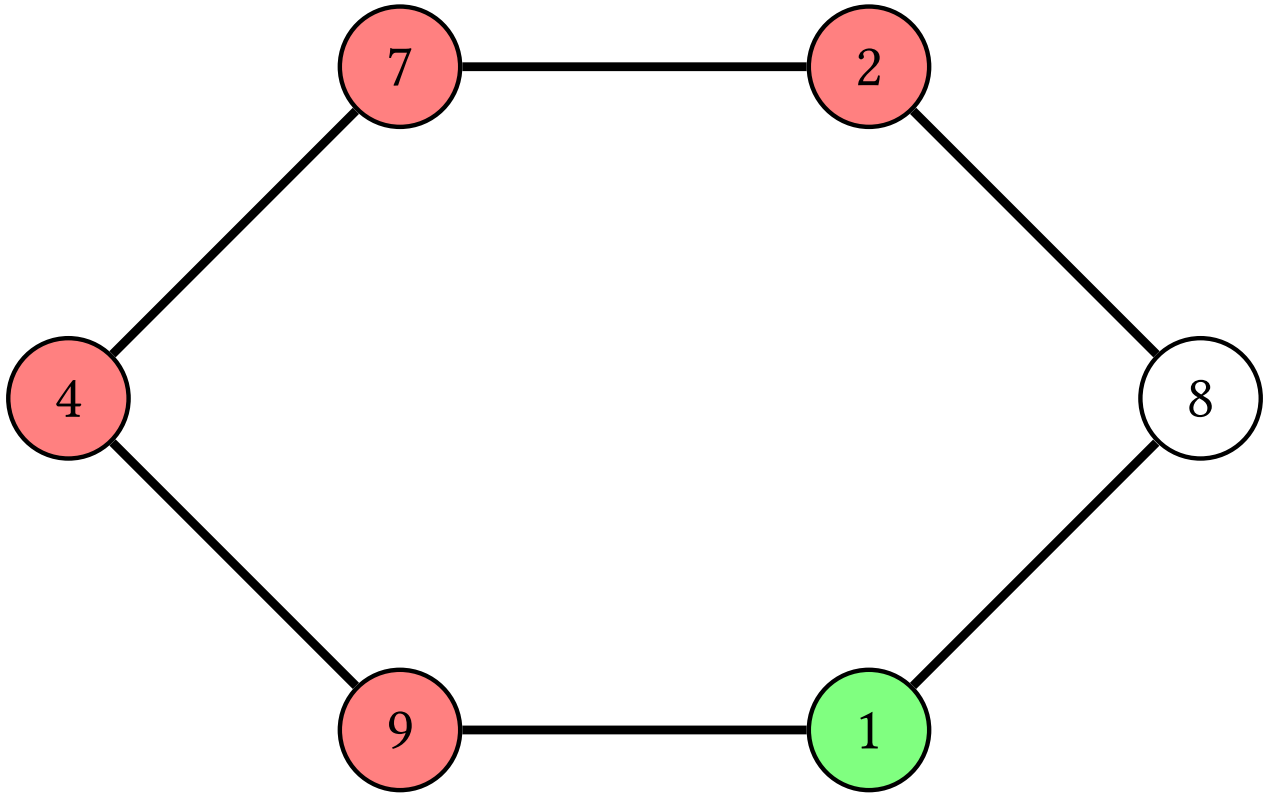
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 10:



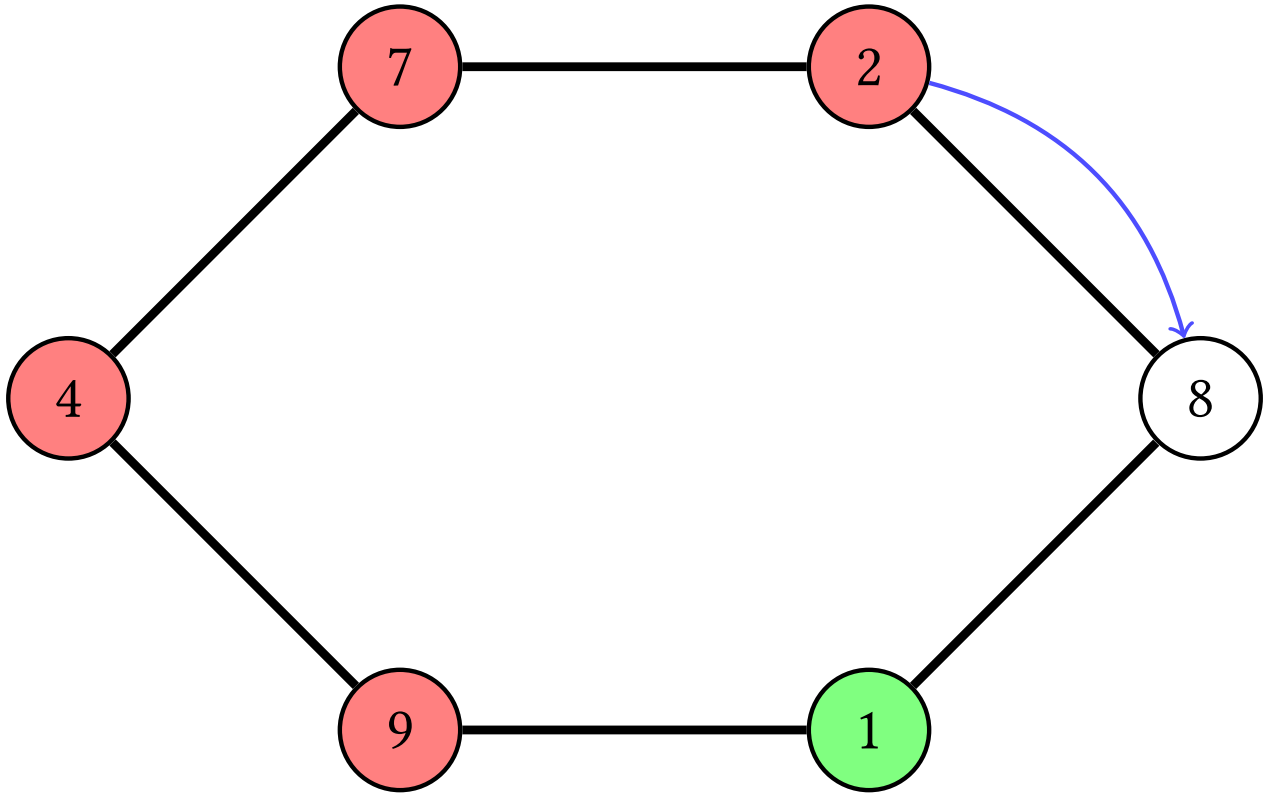
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 11:



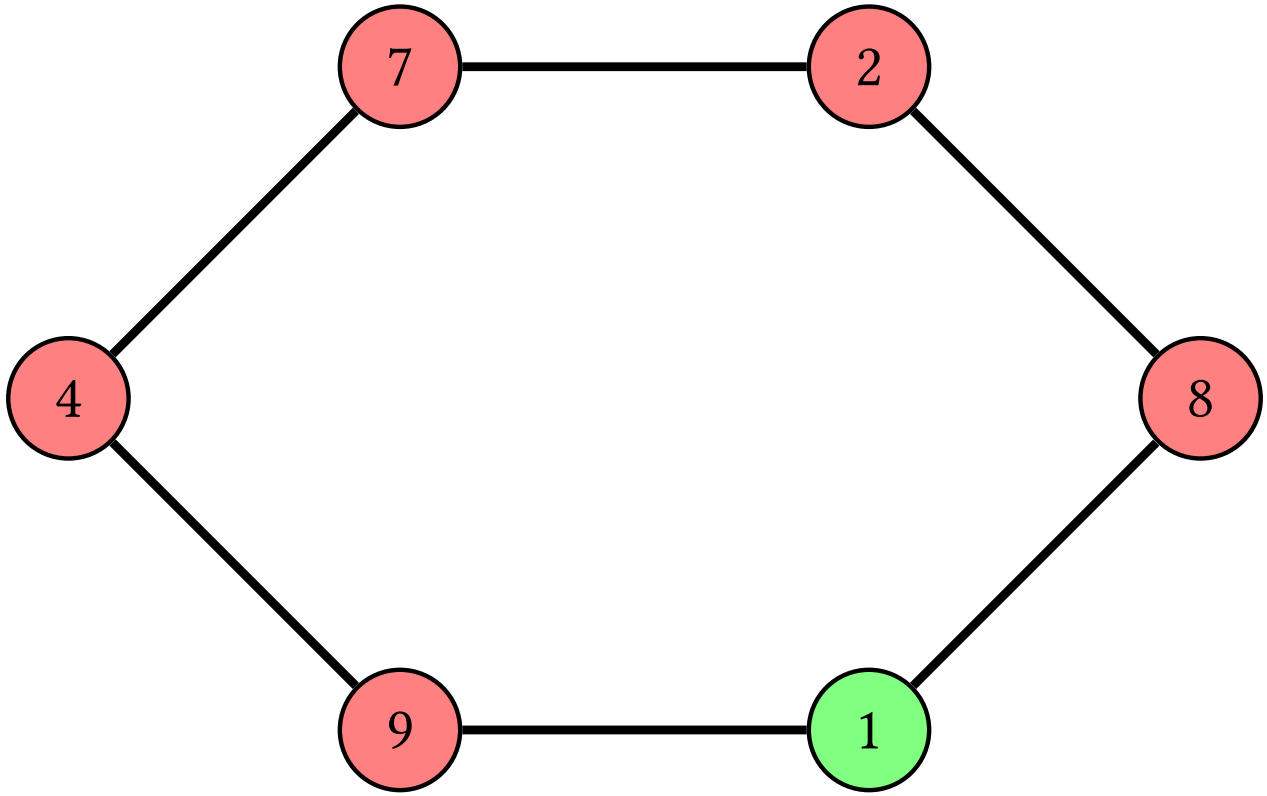
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 11:



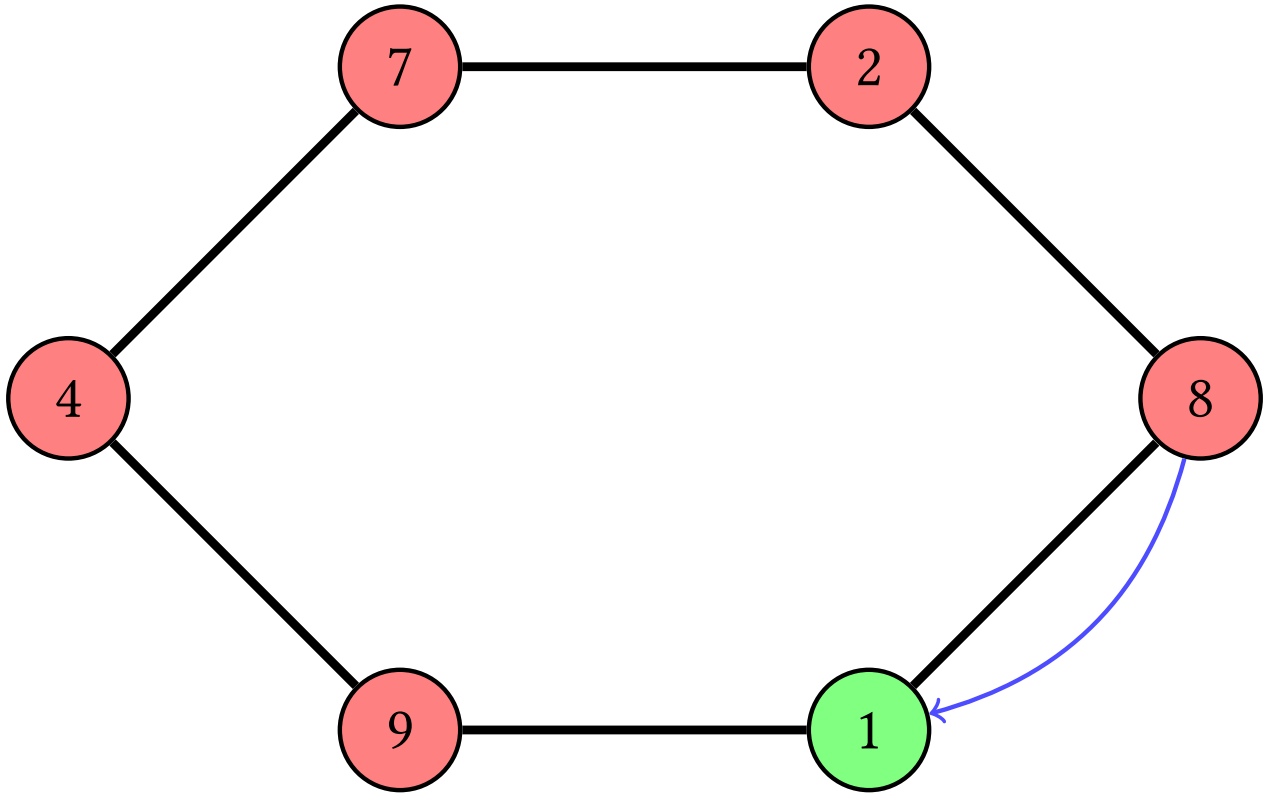
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 12:



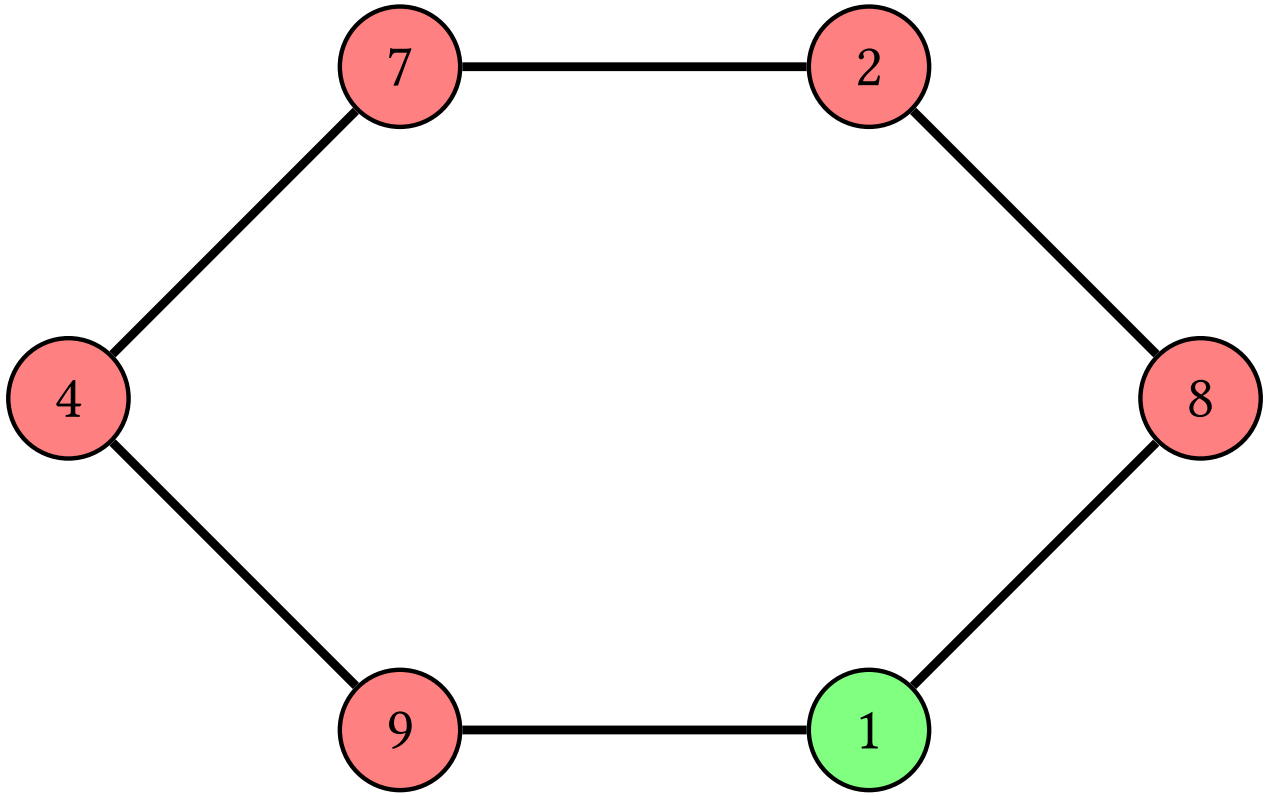
Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 12:



Nachrichteneffizienter Algorithmus (Synchron): Beispiel

Runde 13:



Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.
- Anzahl Nachrichten: $O(n)$

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.
- Anzahl Nachrichten: $O(n)$
- Rundenzahl: $\Theta(n \cdot \min_v \mathrm{ID}(v))$

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.
- Anzahl Nachrichten: $O(n)$
- Rundenzahl: $\Theta(n \cdot \min_v \mathrm{ID}(v))$
  Beispiel: Für IDs im Bereich von 0 bis $2n$ könnte kleinste ID den Wert n haben. Dann werden bereits $\Theta(n^2)$ Runden benötigt.

Zusammenfassung

Leader Election:

1. Unmöglichkeit für deterministische Algorithmen in anonymen Ringen

Zusammenfassung

Leader Election:

1. Unmöglichkeit für deterministische Algorithmen in anonymen Ringen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus

Zusammenfassung

Leader Election:

1. Unmöglichkeit für deterministische Algorithmen in anonymen Ringen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
   - Synchrone und asynchrone Variante

Zusammenfassung

Leader Election:

1. Unmöglichkeit für deterministische Algorithmen in anonymen Ringen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
   - Synchrone und asynchrone Variante
3. Nachrichteneffizienter synchroner Algorithmus:
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten

Zusammenfassung

Leader Election:

1. Unmöglichkeit für deterministische Algorithmen in anonymen Ringen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
   - Synchrone und asynchrone Variante
3. Nachrichteneffizienter synchroner Algorithmus:
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten
   - Frage: Effizienz in beiden Metriken möglich?

Zusammenfassung

Leader Election:

1. Unmöglichkeit für deterministische Algorithmen in anonymen Ringen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
   - Synchrone und asynchrone Variante
3. Nachrichteneffizienter synchroner Algorithmus:
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten
   - Frage: Effizienz in beiden Metriken möglich?

   Nächste Vorlesungseinheit: $O(n)$ Runden, $O(n \log n)$ Nachrichten

Anmerkungen

Das Konzept dieser Vorlesungseinheit wurde zum Teil von Stefan Schmid und Robert Elsässer entwickelt.

Literatur: Nancy A. Lynch: Distributed Algorithms, Kapitel 3. Morgan Kaufmann, 1996.