Synchronisierung

Algorithmen für verteilte Systeme

Sebastian Forster

Universität Salzburg

Synchron vs. asynchron

CONGEST Modell:

- Netzwerk mit n Knoten und m Kanten
- Kommunikation mit Nachbarn in synchronen Runden
- Bandbreite (= maximale Nachrichtengröße) $O(\log n)$
- Zahlreiche Algorithmen für klassische Graphprobleme

Synchron vs. asynchron

CONGEST Modell:

- Netzwerk mit n Knoten und m Kanten
- Kommunikation mit Nachbarn in synchronen Runden
- Bandbreite (= maximale Nachrichtengröße) $O(\log n)$
- Zahlreiche Algorithmen für klassische Graphprobleme

Aber:

- Annahme synchroner Runden ist idealisiert
- In realen Anwendungen können unterschiedlich lange Delays Asynchronizität hervorrufen

Synchron vs. asynchron

CONGEST Modell:

- Netzwerk mit n Knoten und m Kanten
- Kommunikation mit Nachbarn in synchronen Runden
- Bandbreite (= maximale Nachrichtengröße) $O(\log n)$
- Zahlreiche Algorithmen für klassische Graphprobleme

Aber:

- Annahme synchroner Runden ist idealisiert
- In realen Anwendungen können unterschiedlich lange Delays Asynchronizität hervorrufen

Ziel

Simuliere synchrone Algorithmen im asynchronen Modell mit moderaten Overheads in der Zeit- und Nachrichtenkomplexität

Asynchrones Modell

- Event-basiert statt diskrete Zeitschritte

Asynchrones Modell

- Event-basiert statt diskrete Zeitschritte
- Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

Asynchrones Modell

- Event-basiert statt diskrete Zeitschritte
- Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht
- Jede Nachrichtenübermittlung benötigt endliche Zeit

Asynchrones Modell

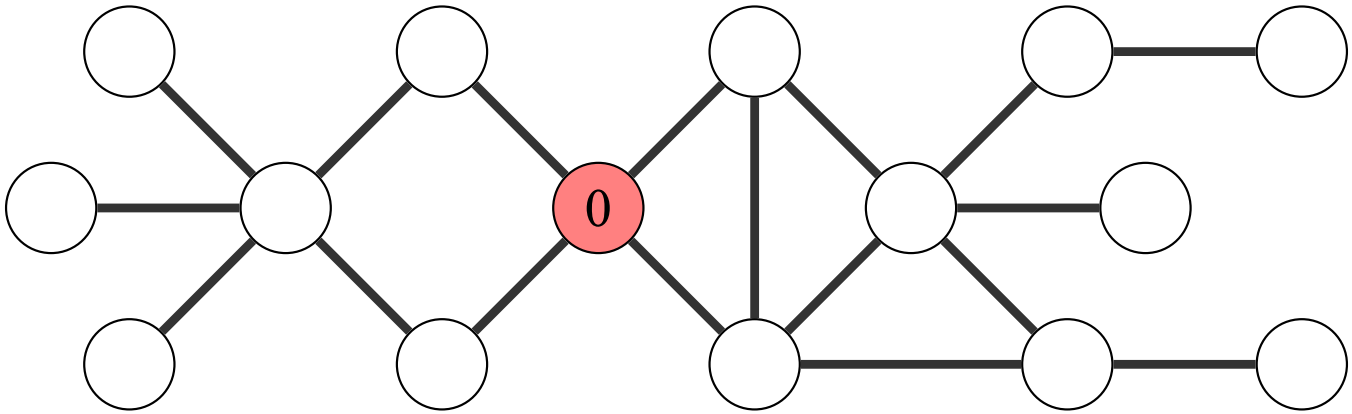
- Event-basiert statt diskrete Zeitschritte
- Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht
- Jede Nachrichtenübermittlung benötigt endliche Zeit
- Für Laufzeitanalyse: Delay von höchstens einer Zeiteinheit

Asynchrones Modell

- Event-basiert statt diskrete Zeitschritte
- Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht
- Jede Nachrichtenübermittlung benötigt endliche Zeit
- Für Laufzeitanalyse: Delay von höchstens einer Zeiteinheit
- Korrektheit muss für beliebige Delays gelten

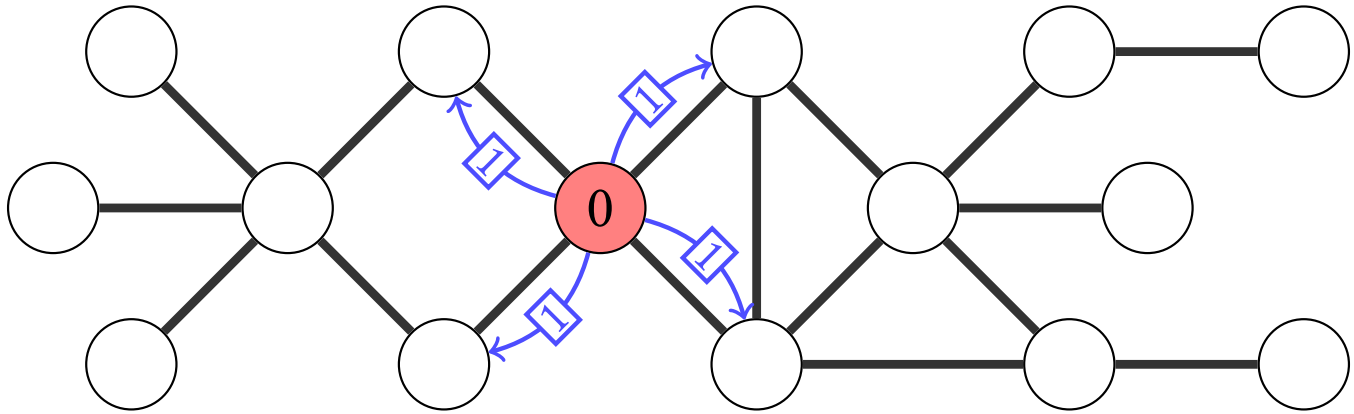
Beispiel: Breitensuche

Naive Ausführung synchroner Algorithmen ist in der Regel nicht erfolgreich (Inkonsistenzen)



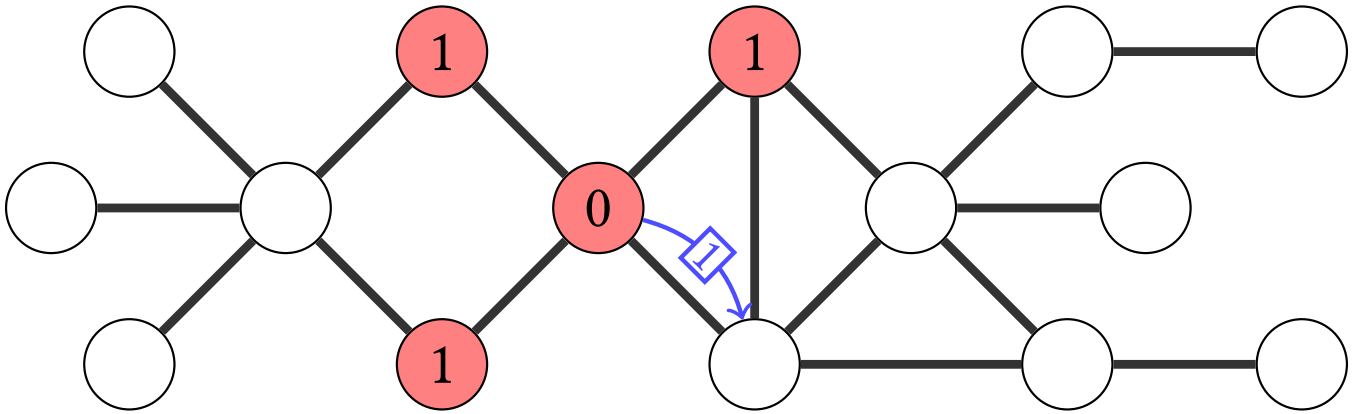
Beispiel: Breitensuche

Naive Ausführung synchroner Algorithmen ist in der Regel nicht erfolgreich (Inkonsistenzen)



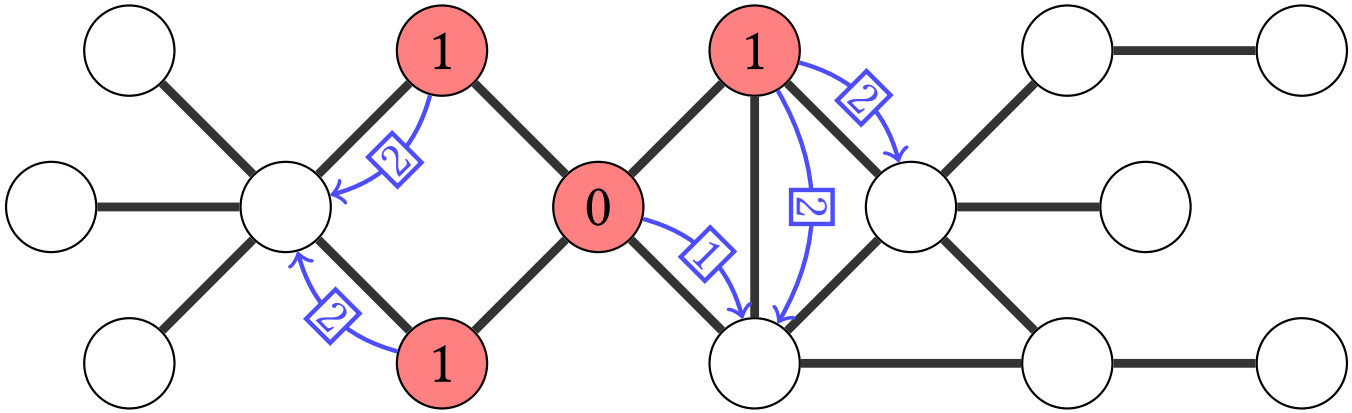
Beispiel: Breitensuche

Naive Ausführung synchroner Algorithmen ist in der Regel nicht erfolgreich (Inkonsistenzen)



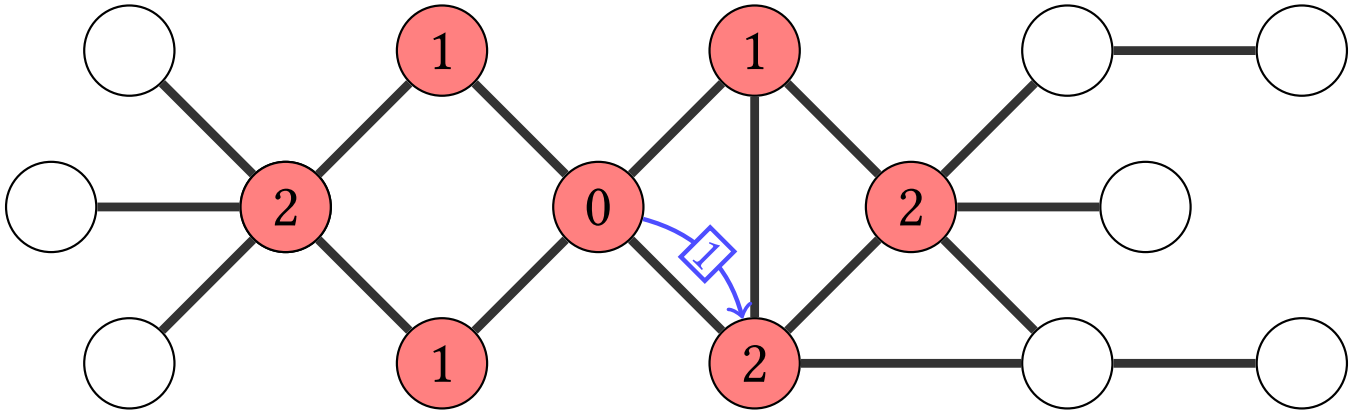
Beispiel: Breitensuche

Naive Ausführung synchroner Algorithmen ist in der Regel nicht erfolgreich (Inkonsistenzen)



Beispiel: Breitensuche

Naive Ausführung synchroner Algorithmen ist in der Regel nicht erfolgreich (Inkonsistenzen)



Pulsgeber

Ziel: Simuliere synchronen Algorithmus in asynchronem Netzwerk

Pulsgeber

Ziel: Simuliere synchronen Algorithmus in asynchronem Netzwerk

Zusätzlicher Pulsgeber-Algorithmus (**Synchronizer**)

Pulsgeber

Ziel: Simuliere synchronen Algorithmus in asynchronem Netzwerk

Zusätzlicher Pulsgeber-Algorithmus (**Synchronizer**)

- Generiert lokalen Puls für jeden Knoten
- Beim Puls i eines Knotens führt dieser Runde i aus: verarbeite alle bisher empfangenen Nachrichten und sende Nachrichten an Nachbarn
- Simulation ist korrekt wenn Puls $i + 1$ immer erst dann generiert wird, wenn Nachrichten von allen Nachbarn für Puls i empfangen wurden
- Achtung: Puls selbst ist nicht synchron für verschiedene Knoten

Pulsgeber

Ziel: Simuliere synchronen Algorithmus in asynchronem Netzwerk

Zusätzlicher Pulsgeber-Algorithmus (**Synchronizer**)

- Generiert lokalen Puls für jeden Knoten
- Beim Puls i eines Knotens führt dieser Runde i aus: verarbeite alle bisher empfangenen Nachrichten und sende Nachrichten an Nachbarn
- Simulation ist korrekt wenn Puls $i + 1$ immer erst dann generiert wird, wenn Nachrichten von allen Nachbarn für Puls i empfangen wurden
- Achtung: Puls selbst ist nicht synchron für verschiedene Knoten

Schwierigkeit: Knoten weiß nicht, von welchen Nachbarn er Nachrichten erhalten müsste

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben.

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben. (Intuition: Simulation von Runde i ist für Knoten komplett abgeschlossen.)

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben. (Intuition: Simulation von Runde i ist für Knoten komplett abgeschlossen.)

Überprüfen der Safety:

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben. (Intuition: Simulation von Runde i ist für Knoten komplett abgeschlossen.)

Überprüfen der Safety:

- Nach dem Empfang jeder Nachricht wird Bestätigungsnachricht an den jeweiligen Nachbarn gesendet

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben. (Intuition: Simulation von Runde i ist für Knoten komplett abgeschlossen.)

Überprüfen der Safety:

- Nach dem Empfang jeder Nachricht wird Bestätigungsnachricht an den jeweiligen Nachbarn gesendet
- Zeit- und Nachrichtenkomplexität verdoppeln sich dadurch

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben. (Intuition: Simulation von Runde i ist für Knoten komplett abgeschlossen.)

Überprüfen der Safety:

- Nach dem Empfang jeder Nachricht wird Bestätigungsnachricht an den jeweiligen Nachbarn gesendet
- Zeit- und Nachrichtenkomplexität verdoppeln sich dadurch

Beobachtung

Wenn für jeden Knoten immer Puls $i + 1$ erst dann generiert wird, sobald Knoten selbst und alle Nachbarn safe für Puls i sind, dann wird synchroner Algorithmus korrekt simuliert.

Safety

Definition

Ein Knoten ist *safe* für Puls i, wenn alle seine in der Simulation von Runde i gesendeten Nachrichten die jeweiligen Nachbarn erreicht haben. (Intuition: Simulation von Runde i ist für Knoten komplett abgeschlossen.)

Überprüfen der Safety:

- Nach dem Empfang jeder Nachricht wird Bestätigungsnachricht an den jeweiligen Nachbarn gesendet
- Zeit- und Nachrichtenkomplexität verdoppeln sich dadurch

Beobachtung

Wenn für jeden Knoten immer Puls $i + 1$ erst dann generiert wird, sobald Knoten selbst und alle Nachbarn safe für Puls i sind, dann wird synchroner Algorithmus korrekt simuliert.

$\rightarrow$ Unterschiedliche Strategien, um festzustellen, ob alle Nachbarn safe sind

Synchronizer α

Algorithmus:

- Sobald Knoten safe ist, sendet er eine safe-Nachricht an alle Nachbarn
- Knoten erkennen dadurch sofort, wenn alle Nachbarn safe sind

Synchronizer α

Algorithmus:

- Sobald Knoten safe ist, sendet er eine safe-Nachricht an alle Nachbarn
- Knoten erkennen dadurch sofort, wenn alle Nachbarn safe sind

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R)$ mit $O(M + Rm)$ Nachrichten simuliert werden.

Synchronizer α

Algorithmus:

- Sobald Knoten safe ist, sendet er eine safe-Nachricht an alle Nachbarn
- Knoten erkennen dadurch sofort, wenn alle Nachbarn safe sind

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R)$ mit $O(M + Rm)$ Nachrichten simuliert werden.

Vorteil: Rein lokal, ohne Overhead in der Zeitkomplexität

Synchronizer α

Algorithmus:

- Sobald Knoten safe ist, sendet er eine safe-Nachricht an alle Nachbarn
- Knoten erkennen dadurch sofort, wenn alle Nachbarn safe sind

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R)$ mit $O(M + Rm)$ Nachrichten simuliert werden.

Vorteil: Rein lokal, ohne Overhead in der Zeitkomplexität

Nachteil: Overhead in der Nachrichtenkomplexität

Synchronizer β

Idee: Puls wird global vom Leader generiert

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel
- Jeder Puls wird per Downcast im Baum an alle Knoten gesendet

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel
- Jeder Puls wird per Downcast im Baum an alle Knoten gesendet
- Beim Erhalt von Puls i führen Knoten Runde i aus

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel
- Jeder Puls wird per Downcast im Baum an alle Knoten gesendet
- Beim Erhalt von Puls i führen Knoten Runde i aus
- Knoten ist safe, sobald er alle Empfangsbestätigungen erhalten hat

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel
- Jeder Puls wird per Downcast im Baum an alle Knoten gesendet
- Beim Erhalt von Puls i führen Knoten Runde i aus
- Knoten ist safe, sobald er alle Empfangsbestätigungen erhalten hat
- Neuer Puls wird von Leader generiert sobald alle Knoten safe sind

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel
- Jeder Puls wird per Downcast im Baum an alle Knoten gesendet
- Beim Erhalt von Puls i führen Knoten Runde i aus
- Knoten ist safe, sobald er alle Empfangsbestätigungen erhalten hat
- Neuer Puls wird von Leader generiert sobald alle Knoten safe sind
- Leader erfährt durch aggregierten Upcast im Baum, wenn alle Knoten safe sind

Synchronizer β

Idee: Puls wird global vom Leader generiert

- Breitensuchbaum mit Leader als Wurzel
- Jeder Puls wird per Downcast im Baum an alle Knoten gesendet
- Beim Erhalt von Puls i führen Knoten Runde i aus
- Knoten ist safe, sobald er alle Empfangsbestätigungen erhalten hat
- Neuer Puls wird von Leader generiert sobald alle Knoten safe sind
- Leader erfährt durch aggregierten Upcast im Baum, wenn alle Knoten safe sind

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(RD)$ mit $O(M + Rn)$ Nachrichten simuliert werden, wenn bereits ein Leader und ein Breitensuchbaum berechnet wurden.

Asynchrone Breitensuche

Achtung: Synchronizer β benötigt Breitensuchbaum

Asynchrone Breitensuche

Achtung: Synchronizer β benötigt Breitensuchbaum

Asynchrone Berechnung eines Breitensuchbaums:

- Synchroner Algorithmus: $O(D)$ Runden, $O(m)$ Nachrichten
  Mit Synchronizer α: $O(D)$ Zeiteinheiten, $O(mD)$ Nachrichten
- Alternative: $O(D^2)$ Zeiteinheiten, $O(m + nD)$ Nachrichten

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Definition

Ein hierarchisches (ρ, μ, ℓ)-Cover ist eine eine Menge von Clusterings $C_1, \ldots C_\ell$ mit folgenden Eigenschaften:

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Definition

Ein hierarchisches (ρ, μ, ℓ)-Cover ist eine eine Menge von Clusterings $C_1, \ldots C_\ell$ mit folgenden Eigenschaften:

1. In jedem Clustering C_i sind die Cluster nicht überlappend.

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Definition

Ein hierarchisches (ρ, μ, ℓ)-Cover ist eine eine Menge von Clusterings $C_1, \ldots C_\ell$ mit folgenden Eigenschaften:

1. In jedem Clustering C_i sind die Cluster nicht überlappend.
2. Jeder Knoten ist in mindestens einem Cluster enthalten.

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Definition

Ein hierarchisches (ρ, μ, ℓ)-Cover ist eine eine Menge von Clusterings $C_1, \ldots C_\ell$ mit folgenden Eigenschaften:

1. In jedem Clustering C_i sind die Cluster nicht überlappend.
2. Jeder Knoten ist in mindestens einem Cluster enthalten.
3. Jedes Cluster hat ein designiertes Zentrum und alle Knoten des Clusters sind einem Baum mit Entfernung höchstens ρ vom Zentrum organisiert.

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Definition

Ein hierarchisches (ρ, μ, ℓ)-Cover ist eine eine Menge von Clusterings $C_1, \ldots C_\ell$ mit folgenden Eigenschaften:

1. In jedem Clustering C_i sind die Cluster nicht überlappend.
2. Jeder Knoten ist in mindestens einem Cluster enthalten.
3. Jedes Cluster hat ein designiertes Zentrum und alle Knoten des Clusters sind einem Baum mit Entfernung höchstens ρ vom Zentrum organisiert.
4. Über alle Knoten gerechnet ist die Gesamtzahl benachbarter Cluster aus einem Clustering mit gleichem oder höherem Index höchstens μ.

Hybride Synchronisierung

Idee: Hybrid zwischen lokalem und globalem Puls

Definition

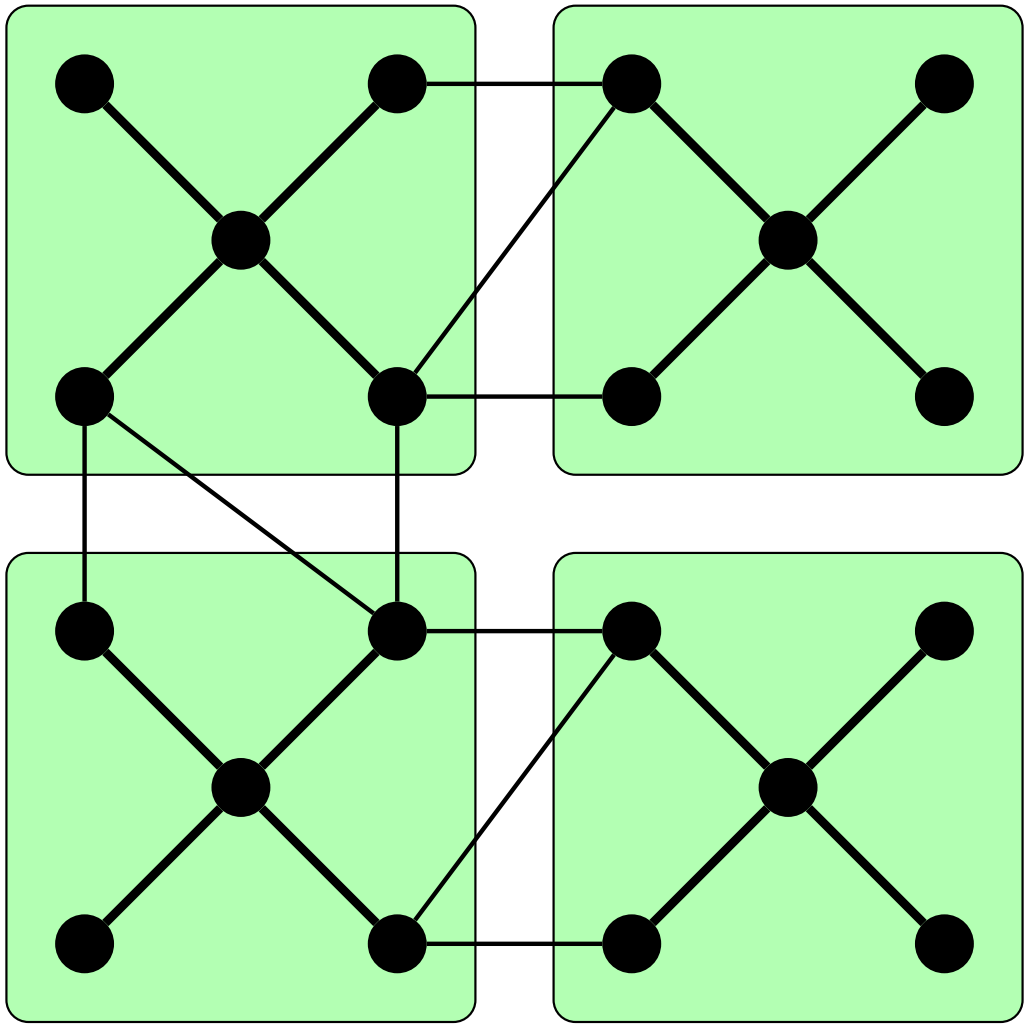
Ein hierarchisches (ρ, μ, ℓ)-Cover ist eine eine Menge von Clusterings $C_1, \ldots C_\ell$ mit folgenden Eigenschaften:

1. In jedem Clustering C_i sind die Cluster nicht überlappend.
2. Jeder Knoten ist in mindestens einem Cluster enthalten.
3. Jedes Cluster hat ein designiertes Zentrum und alle Knoten des Clusters sind einem Baum mit Entfernung höchstens ρ vom Zentrum organisiert.
4. Über alle Knoten gerechnet ist die Gesamtzahl benachbarter Cluster aus einem Clustering mit gleichem oder höherem Index höchstens μ.

Vorgehensweise:

- Synchronizer α zwischen den Clustern
- Synchronizer β innerhalb der Cluster

Beispiel für Clustering



Berechnung eines hierarchischen Covers

Idee: Verwende Zwischenergebnis des verteilten Spanner-Algorithmus von Baswana und Sen

Berechnung eines hierarchischen Covers

Idee: Verwende Zwischenergebnis des verteilten Spanner-Algorithmus von Baswana und Sen

Theorem ([Baswana/Sen '03])

Im synchronen CONGEST Modell kann für jede Ganzzahl $k \geq 2$ ein hierarchisches ein (ρ, μ, ℓ)-Cover mit $\rho = k - 1$, $\mu = O(n^{1+1/k})$ in Erwartung und $\ell = k$ in $O(k^2)$ Runden berechnet werden.

Berechnung eines hierarchischen Covers

Idee: Verwende Zwischenergebnis des verteilten Spanner-Algorithmus von Baswana und Sen

Theorem ([Baswana/Sen '03])

Im synchronen CONGEST Modell kann für jede Ganzzahl $k \geq 2$ ein hierarchisches ein (ρ, μ, ℓ)-Cover mit $\rho = k - 1$, $\mu = O(n^{1+1/k})$ in Erwartung und $\ell = k$ in $O(k^2)$ Runden berechnet werden.

Mit Synchronizer α:

Corollary

Im asynchronen CONGEST Modell kann für jede Ganzzahl $k \geq 2$ ein hierarchisches (ρ, μ, ℓ)-Cover mit $\rho = k - 1$, $\mu = O(n^{1+1/k})$ in Erwartung und $\ell = k$ in Zeit $O(k^2)$ mit $O(mk^2)$ Nachrichten berechnet werden.

Synchronizer γ

Algorithmus:

- Synchronisierung innerhalb eines Clusters wie bei Synchronizer β:

Synchronizer γ

Algorithmus:

- Synchronisierung innerhalb eines Clusters wie bei Synchronizer β:
  - Jedes Zentrum eines Clusters generiert Puls für alle Knoten des Clusters
  - Downcast des Pulses im Baum
  - Knoten ist safe sobald er alle Empfangsbestätigungen erhalten hat
  - Cluster ist safe, wenn alle Knoten des Clusters safe sind
  - Zentrum erfährt durch aggregierten Upcast im Baum, wenn Cluster safe ist

Synchronizer γ

Algorithmus:

- Synchronisierung innerhalb eines Clusters wie bei Synchronizer β:
  - Jedes Zentrum eines Clusters generiert Puls für alle Knoten des Clusters
  - Downcast des Pulses im Baum
  - Knoten ist safe sobald er alle Empfangsbestätigungen erhalten hat
  - Cluster ist safe, wenn alle Knoten des Clusters safe sind
  - Zentrum erfährt durch aggregierten Upcast im Baum, wenn Cluster safe ist
- Synchronisierung zwischen den Clustern wie bei Synchronizer α:

Synchronizer γ

Algorithmus:

- Synchronisierung innerhalb eines Clusters wie bei Synchronizer β:
  - Jedes Zentrum eines Clusters generiert Puls für alle Knoten des Clusters
  - Downcast des Pulses im Baum
  - Knoten ist safe sobald er alle Empfangsbestätigungen erhalten hat
  - Cluster ist safe, wenn alle Knoten des Clusters safe sind
  - Zentrum erfährt durch aggregierten Upcast im Baum, wenn Cluster safe ist
- Synchronisierung zwischen den Clustern wie bei Synchronizer α:
  - Sobald Cluster safe ist, werden benachbarte Cluster darüber informiert:
    - ★ Downcast der Information im eigenen Cluster
    - ★ Knoten im eigenen Cluster informieren benachbarte Cluster über ausgewählte Kanten zu Nachbarn in diesen Clustern
  - Zentrum erfährt durch aggregierten Upcast im Baum, wenn alle Nachbarcluster safe sind
  - Sobald alle Nachbarcluster safe sind, wird neuer Puls generiert

Details

- Jeder Knoten stellt für jedes benachbarte Cluster C aus Clustering mit gleichem oder höherem Index **eine** Verbindung zu einem Nachbarknoten aus C her

Details

- Jeder Knoten stellt für jedes benachbarte Cluster C aus Clustering mit gleichem oder höherem Index **eine** Verbindung zu einem Nachbarknoten aus C her

  Insgesamt werden also μ Verbindungen hergestellt

Details

- Jeder Knoten stellt für jedes benachbarte Cluster C aus Clustering mit gleichem oder höherem Index **eine** Verbindung zu einem Nachbarknoten aus C her

  Insgesamt werden also μ Verbindungen hergestellt
- Für jeden Knoten sind die ausgewählten Kante zu Nachbarclustern jene, mit denen er eine Verbindung in ein Nachbarcluster hergestellt hat, sowie jene, mit denen zu ihm eine Verbindung aus einem Nachbarcluster hergestellt wurde

Details

- Jeder Knoten stellt für jedes benachbarte Cluster C aus Clustering mit gleichem oder höherem Index **eine** Verbindung zu einem Nachbarknoten aus C her

  Insgesamt werden also μ Verbindungen hergestellt
- Für jeden Knoten sind die ausgewählten Kante zu Nachbarclustern jene, mit denen er eine Verbindung in ein Nachbarcluster hergestellt hat, sowie jene, mit denen zu ihm eine Verbindung aus einem Nachbarcluster hergestellt wurde
- Kommunikation mit Nachbarclustern (Austausch von Informationen über Safety der Cluster) erfolgt über ausgewählte Kanten

Details

- Jeder Knoten stellt für jedes benachbarte Cluster C aus Clustering mit gleichem oder höherem Index **eine** Verbindung zu einem Nachbarknoten aus C her

  Insgesamt werden also μ Verbindungen hergestellt
- Für jeden Knoten sind die ausgewählten Kante zu Nachbarclustern jene, mit denen er eine Verbindung in ein Nachbarcluster hergestellt hat, sowie jene, mit denen zu ihm eine Verbindung aus einem Nachbarcluster hergestellt wurde
- Kommunikation mit Nachbarclustern (Austausch von Informationen über Safety der Cluster) erfolgt über ausgewählte Kanten
- Jeder Knoten ist in höchstens ℓ Clustern enthalten und kann unter Umständen zu jedem Zeitpunkt nur für höchstens eines der Cluster Downcast oder Upcast-Nachrichten senden

Details

- Jeder Knoten stellt für jedes benachbarte Cluster C aus Clustering mit gleichem oder höherem Index **eine** Verbindung zu einem Nachbarknoten aus C her

  Insgesamt werden also μ Verbindungen hergestellt
- Für jeden Knoten sind die ausgewählten Kante zu Nachbarclustern jene, mit denen er eine Verbindung in ein Nachbarcluster hergestellt hat, sowie jene, mit denen zu ihm eine Verbindung aus einem Nachbarcluster hergestellt wurde
- Kommunikation mit Nachbarclustern (Austausch von Informationen über Safety der Cluster) erfolgt über ausgewählte Kanten
- Jeder Knoten ist in höchstens ℓ Clustern enthalten und kann unter Umständen zu jedem Zeitpunkt nur für höchstens eines der Cluster Downcast oder Upcast-Nachrichten senden
- Knoten generiert neuen Puls, wenn er von allen Cluster-Zentren Puls-Nachricht erhalten hat

Garantien

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R\ell\rho)$ mit $O(M + R(\mu + \ell n))$ Nachrichten simuliert werden, wenn bereits ein hierarchisches (ρ, μ, ℓ)-Cover berechnet wurde.

Garantien

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R\ell\rho)$ mit $O(M + R(\mu + \ell n))$ Nachrichten simuliert werden, wenn bereits ein hierarchisches (ρ, μ, ℓ)-Cover berechnet wurde.

Mit Algorithmus zur Berechnung eines hierarchischen Covers:

Corollary

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(Rk^2)$ mit $O(M + R(n^{1+1/k} + kn) + mk^2)$ Nachrichten simuliert werden.

Garantien

Theorem

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R\ell\rho)$ mit $O(M + R(\mu + \ell n))$ Nachrichten simuliert werden, wenn bereits ein hierarchisches (ρ, μ, ℓ)-Cover berechnet wurde.

Mit Algorithmus zur Berechnung eines hierarchischen Covers:

Corollary

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(Rk^2)$ mit $O(M + R(n^{1+1/k} + kn) + mk^2)$ Nachrichten simuliert werden.

Corollary

Jeder synchrone Algorithmus, der R Runden benötigt und dabei M Nachrichten versendet, kann im asynchronen Modell in Zeit $O(R \log^2 n)$ mit $O(M + Rn \log n + m \log^2 n)$ Nachrichten simuliert werden.

Zusammenfassung

	Zeit	**Nachrichten**	**Initialisierungsaufwand**
Synchronizer α	$O(R)$	$O(M + Rm)$	–
Synchronizer β	$O(RD)$	$O(M + Rn)$	Breitensuchbaum
Synchronizer γ	$O(R\ell\rho)$	$O(M + R(\mu + \ell n))$	(ρ, μ, ℓ)-Cover

Zusammenfassung

	Zeit	**Nachrichten**	**Initialisierungsaufwand**
Synchronizer α	$O(R)$	$O(M + Rm)$	–
Synchronizer β	$O(RD)$	$O(M + Rn)$	Breitensuchbaum
Synchronizer γ	$O(R\ell\rho)$	$O(M + R(\mu + \ell n))$	(ρ, μ, ℓ)-Cover

Rechtfertigt Einschränkung auf synchrone Algorithmen!

Zusammenfassung

	Zeit	**Nachrichten**	**Initialisierungsaufwand**
Synchronizer α	$O(R)$	$O(M + Rm)$	–
Synchronizer β	$O(RD)$	$O(M + Rn)$	Breitensuchbaum
Synchronizer γ	$O(R\ell\rho)$	$O(M + R(\mu + \ell n))$	(ρ, μ, ℓ)-Cover

Rechtfertigt Einschränkung auf synchrone Algorithmen!

Überlappung zwischen folgenden Themen:

- Synchronisierung
- Spanner
- Cover und Partitionierungen

Quellen

Der Inhalt dieser Vorlesungseinheit basiert zum Teil auf einer Vorlesungseinheit von Christoph Lenzen.

Literatur:

- Baruch Awerbuch. „Complexity of Network Synchronization“. *Journal of the ACM* 32(4): 804–823 (1985)
- Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, Michael E. Saks. „Adapting to Asynchronous Dynamic Networks“. In: *Proc. of the Symposium on Theory of Computing (STOC)*. 1992, S. 557–570
- David Peleg (2000) *Distributed Computing*, Kapitel 6, SIAM.