Algorithmen für verteilte Systeme **Universität Salzburg**
Sommersemester 2018 **FB Computerwissenschaften**

Vorlesung 4: 7. Mai 2018

Vortrag: Sebastian Krinninger *Mitschrift: Falco Gödde*

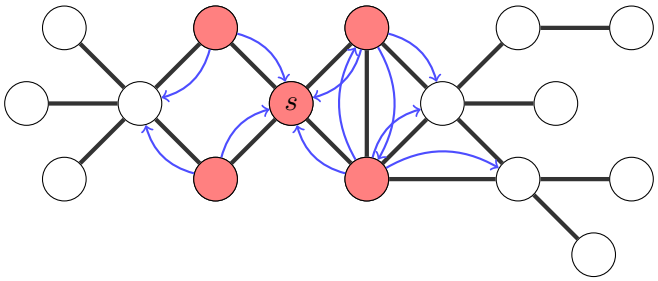
4.1 Einführung in das CONGEST Modell

Im CONGEST Modell wird ein ungerichteter und verbundener Graph modelliert, der gewichtet oder ungewichtetet sein kann. Die Kommunikation, also das Versenden von Nachrichten zwischen den n Knoten und m Kanten, findet in synchronisierten Runden statt, wobei in jeder Runde jeder Knoten pro Nachbar maximal eine Nachricht versenden kann. Außerdem besitzt jeder Knoten im Graph eine eindeutige ihm bekannte ID. Der Durchmesser D eines Graphen ist die Anzahl der Kanten des kürzesten Wegs zwischen den am weitesten entfernten Knoten. Des Weiteren wird im CONGEST Modell die Nachrichtengröße auf $O(\log n)$ Bits limitiert und die ID jedes Knotens ebenfalls auf $O(\log n)$ Bits.

4.1.1 Broadcast

Für den Broadcast Algorithmus wird angenommen, dass es genau einen Startknoten s gibt, der die Information I der Größe $O(\log n)$ an alle Knoten im Graph senden möchte.

```
1 Wenn Information I das erste Mal empfangen wurde:
2   In darauffolgender Runde: Sende I an alle Nachbarn
```



Es beginnt der Startknoten s: Dieser sendet I an alle unmittelbaren Nachbarn, mit denen er verbunden ist. Dieses senden I wiederum an ihre Nachbarn, bis I auch in den von s am weitesten entfernten Knoten angekommen ist.

4.1.1.1 Korrektheit

Die zu zeigende Invariante lautet:

In Runde r erhält jeder Knoten v mit $\text{dist}(s, v) = r$ die Information I.

Durch die r-te Runde erhalten also alle Knoten, die r Kanten vom Startknoten s entfernt sind, die Information I. Der Beweis gliedert sich, wie üblich für eine Invariante, in Initialisierung, Erhaltung und Terminierung.

1. *Initialisierung*

   In der „0-ten“ Runde, d. h. vor dem ersten Senden von Nachrichten ist mit $\text{dist}(s, v) = 0$ nur der Startknoten s betroffen: Hier gilt also die Invariante, da s die Information I zum Ausgangszeitpunkt hat.

2. *Erhaltung:* $r \to r + 1$

   Sei $r + 1$ die „aktuell“ betrachtete Runde und v ein Knoten mit $\text{dist}(s, v) = r + 1$ und einem Vorgänger u auf dem kürzesten Weg von s nach v. Dann gilt für u: $\text{dist}(s, u) = \text{dist}(s, v) - 1 = r$. Was bedeutet, dass u in der vorherigen Runde (Runde: r) I erhalten hat und in der Runde $r + 1$, I an alle Nachbarn weiterleitet, wobei sich v auch in dieser Menge befindet und daher die Information in Runde $r + 1$ erhält.

3. *Terminierung*

   Da das Netzwerk zusammenhängend ist, hat jeder Knoten eine endliche Distanz zu s, was bedeutet, dass jeder Knoten nach einer endlichen Rundenanzahl I erhält.

4.1.1.2 Laufzeit

Der Algorithmus läuft so lange – hat genau so viele Runden – bis genau jeder Knoten die Information erhalten hat. Es muss also der/die am weitesten von s entfernte Knoten, nennen wir ihn v, erreicht werden. Daher gilt im schlechtesten Fall für die maximale Anzahl der Runden: $r_{\max} = \max_v \text{dist}(s, v)$. Die Exzentrizität eines Knotens u ist: $\text{Ex}(u) = \max_v \text{dist}(s, v)$. Also die Anzahl der Kanten des kürzesten Weges zwischen Knoten u und v.

Zusammenfassend lässt sich für Startknoten s, dem/den am weitesten von s entfernten Knoten v und für die maximale Anzahl der Runden feststellen: $r_{\max} = \text{dist}(s, v) \leq D$. Dadurch gilt für die Laufzeit des Broadcast Algorithmus: $O(\text{Ex}(s)) = O(D)$.

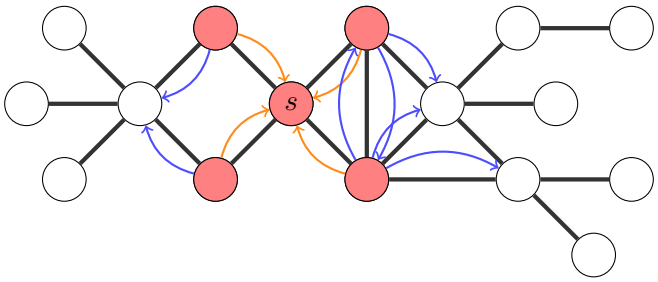
4.1.1.3 Nachrichtenkomplexität

Der Algorithmus bestimmt, dass die Information I nach Erhalt in der darauffolgenden Runde an alle Nachbarn weitergesendet wird. Das heißt, dass jeder Knoten genau so oft I versendet, wie er anliegende Kanten zu seinen Nachbarn hat. Dies entspricht dem sogenannten Grad des Knotens v – also $\text{Grad}(v)$. Insgesamt werden daher: $\sum_v \text{Grad}(v) = O(m)$ viele Nachrichten verschickt; die obere Schranke ergibt sich aus der Tatsache dass die Summe der Knotengrade gleich $2m$ ist.

4.1.2 Spannbaum durch Breitensuche

Das Ziel ist es aus einem beliebigen zusammenhängenden Graphen einen Baum zu gestalten. Es müssen also Eltern-Kind Beziehungen hergestellt werden ohne dass diese im Kreis laufen. Dafür wird eine Information I vom Startknoten s versendet:

```
1 Wenn Information I das erste Mal empfangen wurde:
2   Speichere genau einen der Sender von I als Elternknoten
3   In nächster Runde: Sende I an alle anderen Nachbarn, sende Kind-Nachricht an Elternknoten
```

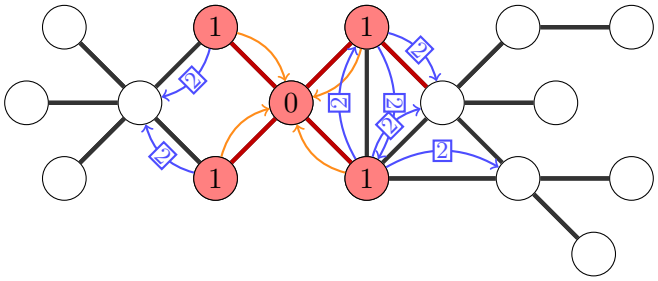


Um für Determinismus zu sorgen kann beispielsweise in Zeile 2 der Knoten, mit der kleineren ID, bevorzugt als Elternknoten gespeichert werden. Nach dem vollständigen Durchlauf des Algorithmus ist ein impliziter Spannbaum entstanden, da jeder Knoten seinen Elternknoten und seine Kinder kennt. Dadurch ist kein Kreis entstanden, da sich alle Knoten (außer s) im Graph für genau einen Elternknoten entschieden haben.

4.1.3 Distanz-Berechnung

Zur Distanz-Berechnung wird der Algorithmus zur Bestimmung eines Spannbaums durch Breitensuche etwas modifiziert bzw. erweitert. Anstelle der Kind-Nachricht senden die Elternknoten ihren Kindern die Distanz, welche sie sich zuweisen sollen.

```
1 Knoten s: Distanz 0, sendet Distanz-Information 1 an alle Nachbarn
2 Knoten v ≠ s: Wenn das erste Mal eine Distanz-Information d empfangen wurde
3   Speichere einen der Sender von d als Elternknoten
4   In nächster Runde: Sende d + 1 an alle anderen Nachbarn, sende Kind-Nachricht an
        Elternknoten
```



Dadurch erhält jeder Knoten im verbundenen Graph einen Elternknoten (außer s) und von genau diesem auch eine Distanz zugeordnet. Falls ein Knoten, der bereits seine Distanz bestimmt hat, eine weitere, höhere erhält, so ignoriert er diese natürlich.

4.1.3.1 Zusammenfassung Breitensuche

Für eine Breitensuche mit Startknoten s gelten folgende Eigenschaften:

- *Anzahl der Runden:* $O(D) = O(\mathrm{Ex}(s))$

  Der Algorithmus terminiert, sobald alle Knoten einen Elternknoten zugewiesen bekommen haben und alle Kind-Nachrichten bei den Elternknoten eingegangen sind. Im worst case ist s D Kanten von einem Blatt entfernt. Daher die Laufzeit $O(D)$, was gleichzeitig die Exzentrizität von s ist.

- *Anzahl der Nachrichten:* $O(m)$

  Es wird über jede Kante mindestens einmal und maximal zweimal eine Nachricht verschickt – also ein (konstantes) Vielfaches von m (maximal: $2m$ Nachrichten), was in $O(m)$ ist.

- *Jeder Knoten v kennt seine Distanz zu s:* $\mathrm{dist}(s, v)$

  Stellt das Ergebnis des Algorithmus zur Distanz-Berechnung dar.

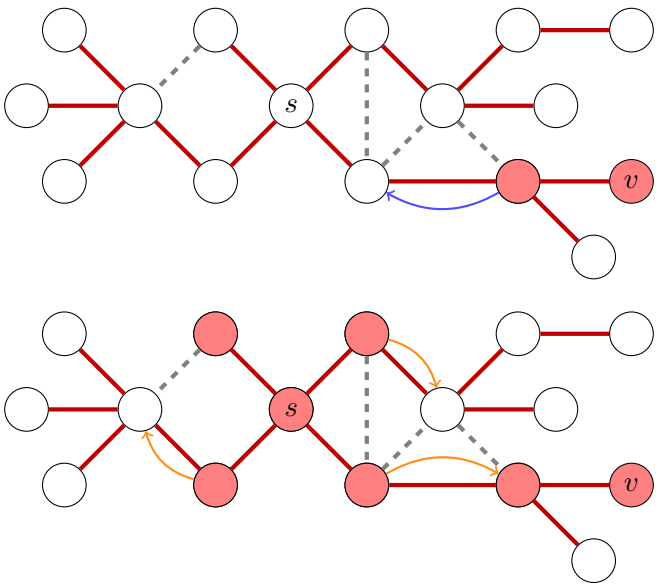
- *Für den Breitensuchbaum gilt*

  - Jeder Knoten $u \neq s$ hat einen Elterknoten v mit $\mathrm{dist}(s, v) = \mathrm{dist}(s, u) - 1$
  - Jeder Knoten kennt seinen Elternknoten und seine Kinder im Baum (jeweils Teilmenge der Nachbarn)
  - Jeder Breitensuchbaum ist auch ein Spannbaum

4.1.4 Upcast/Downcast

Für diese Situation nehmen wir an, dass bereits ein Breitensuchbaum, ausgehend vom Startknoten s, berechnet wurde. Außerdem hat der Knoten v für den Upcast/Downcast eine Information I der Größe $O(\log n)$, die er an alle anderen Knoten im Graph senden möchte.

```
1 Upcast: I wird über Elternknoten zu s gesendet
2 Downcast: I wird über Kinder zu allen Knoten gesendet
```



Beim Upcast sendet der Knoten v also I an seinen Elternknoten und dieser wieder an seinen Elternknoten usw. bis die Information den Startknoten s erreicht hat. Von dort aus wird der Downcast gestartet, indem jeder Knoten I an jedes Kind weiterleitet, bis auch alle Blätter erreicht wurden.

Die Anzahl der Runden entspricht $O(D)$, da im worst case die Knoten v und s, D-weit auseinander sind und diese Distanz für Up- und Downcast insgesamt zweimal zurückgelegt werden muss. Daher $O(2D)$, was in $O(D)$ ist.

Die obere Schranke für die Anzahl der Nachrichten beträgt $O(n)$. Im Gegensatz zum Broadcast, wo über jede Kante im Ursprungsgraphen eine bis zwei Nachrichten gesendet werden, liegt beim Up- bzw. Downcast bereits ein Baum vor, für den folgende Eigenschaft gilt:

$$\text{Anzahl der Knoten} = \text{Anzahl der Kanten} - 1$$

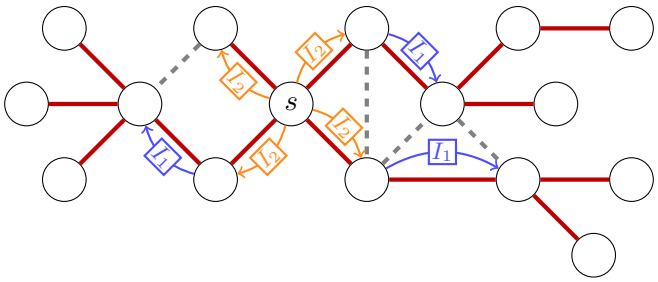
Daher beträgt Anzahl der Nachrichten $O(n)$.

4.1.5 Pipelining

4.1.5.1 Multipler Upcast

Die Ausgangssituation ist wie folgt: Der Startknoten s hat k Informationen $I_1, ..., I_k$ jeweils der Größe $O(\log n)$, wodurch die Informationen nicht zusammenfassen können, da $O(\log n)$ die maximale Nachrichtengröße im CONGEST Modell ist. Das Ziel ist es, allen Knoten im Graph die Informationen $I_1, ..., I_k$ zukommen zu lassen. Ein naiver Ansatz führt einfach k-mal den normalen Broadcast Algorithmus durch, was zur Laufzeit $O(kD)$ führen würde. Allerdings lässt sich diese Laufzeit durch den Pipeline-Algorithmus verringern.

```
1 Wenn eine Information I vom Elternknoten empfangen wurde:
2   In nächster Runde: Sende I an alle Kinder
```



Startknoten s sendet in Runde i Information I_i, bis $i = k$. Alle restlichen Knoten senden in der nächsten Runde jegliche Information an ihre Kinder weiter, sobald sie eine erhalten haben. Dadurch werden die vorhandenen Kanten besser ausgenutzt und die Informationen I_1 bis I_k erreichen früher alle Knoten (als bei k sequentiellen Broadcasts).

Die Invariante lautet:

Nach $r = d + j$ Runden hat jeder Knoten v mit $dist(s, v) = d$ die Informationen $I_1, ..., I_j$ empfangen.

Ein Beispiel zur Veranschaulichung: Nach 4 Runden hat Knoten v mit $\text{dist}(s, v) = 3$ die (erste) Information I_1 erhalten. (Eine Information gilt nach Erhalt erst in der nächsten Runde als empfangen.) In der darauffolgenden Runde erhält er I_2 usw. D. h. in der Runde $3 + j$ erhält er die letzte Information I_j.

Die Laufzeit kann durch das Pipelining von $O(kD)$ auf $O(k + D)$ reduziert werden. $O(D)$ wird benötigt, um den am weitesten entfernten Knoten v (von s aus) zu erreichen und noch einmal k Runden, damit v auch die letzte Information I_k erhalten hat. (Eigentlich sind es $k + 1$ Runden, bis v I_k auch wirklich erhalten hat aber $O(k + 1 + D) = O(k + D)$).

4.1.5.2 Convergecast

Das Ziel von s ist es sicherzustellen, dass der Broadcast bzw. die Breitensuche abgeschlossen ist. Die Lösungsidee beinhaltet das Versenden von ACK Nachrichten (vergleichbar mit TCP) von jedem Knoten an s per Upcast. Eine naive Herangehensweise ist folgende:

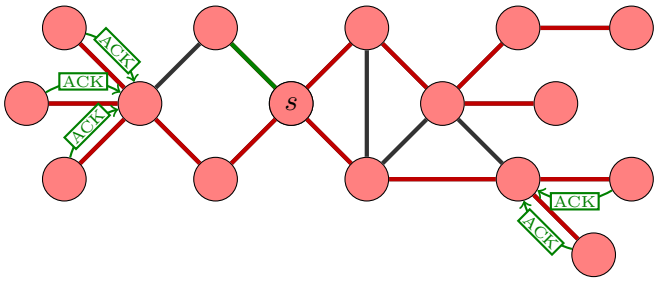
```
1 Führe Breitensuche mit Spannbaumberechnung von Startknoten s aus
2 Jedes Blatt: sende ACK-Nachricht an s per Upcast im Baum
```

Jedoch tritt das Problem auf, dass sich trotz Pipelining an den Elternknoten Stau bilden kann. Im worst case beträgt die Laufzeit $\Theta(n)$.

4.1.5.3 Aggregation der ACK-Nachrichten

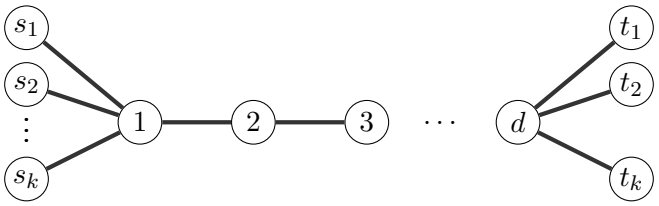
Eine Lösung des Stau-Problems ist die Aggregierung der ACK-Nachrichten, wie folgender Algorithmus beschreibt.

```
1 Führe Breitensuche mit Spannbaumberechnung von Startknoten s aus
2 Sobald Knoten ACK von allen Kindern erhalten: Sende ACK an Elternknoten in nächster Runde
3 Blätter im Baum erkennen sich selbst als solche, wenn sie keine Kind-Nachricht erhalten
4 s wartet bis ACK von allen Kindern erhalten
```



Aus funktionaler Sicht ist das ACK-Bit eines Knotens die Und-Verknüpfung der ACK-Bits seiner Kinder. Es müssen also erst alle ACK-Nachrichten der Kinder eingegangen sein, damit die eigene ACK-Nachricht gebildet und an den Elternknoten abgeschickt werden kann. Durch die Baumstruktur ist die Bottom-Up Berechnung und das Warten auf die Funktionswerte der Kinder möglich. Weitere Aggregatfunktionen sind beispielsweise: Summe, Minimum, Maximum.

4.1.5.4 Theoretische Limits



Die Paare (s_1, t_1) werden als insgesamt k Knotenpaare gesehen. Ziel ist es nun, die Information I_j von s_j an t_j weiterzuleiten. Dabei entstehen die folgenden Probleme:

- Jede Information muss über den Pfad der Länge d gesendet werden, was bedeutet, dass bereit für nur eine Information mindestens d-Runden benötigt werden, um diese über diesen Pfad weiterzuleiten. (Die maximale Pfadlänge wird auch oft als *Dilation* bezeichnet.)
- Jede Information muss über die Kante $(1, 2)$ gesendet werden, wodurch es zu Stau kommen kann und Knoten 1 daher eine Message Queue benötigt. Nur um k Nachrichten von Knoten 1 aus der Queue abzuschicken, werden bereits k Runden benötigt. (Die maximale Anzahl Nachrichten über eine Kante wird auch oft als **Congestion** bezeichnet.)

Jedoch kann trotz dieser Limitierungen oft eine passende obere Schranke $O(\textit{Dilation} + \text{Congestion})$ erreicht werden!