

Obere und untere Schranken beim Broadcasting in Funknetzwerken

Tobias Bräutigam
Burgstraße 31
59581 Warstein-Hirschberg
Mat.-Nr.: 6015207

16. Januar 2005

Zusammenfassung

Diese Arbeit untersucht die Zeit, die eine Nachricht ausgehend von einer Quelle benötigt um alle Teilnehmer eines Multi-Hop Funknetzwerks zu erreichen (Broadcast). Dabei wird davon ausgegangen, dass die Topologie der Netzwerke nicht bekannt ist und es in der Regel keinen direkten Weg (1-Hop) von der Quelle zu allen anderen Knoten gibt. Außerdem wird angenommen, dass ein Knoten nur dann eine Nachricht erhält wenn genau einer seiner Nachbarn diese Nachricht abschickt. Bei mehreren Sender kommt es zu einer Kollision. Ebenfalls wird angenommen, dass die Knoten solche Kollisionen nicht erkennen können, d.h. sie können den Fall das kein Nachbar sendet nicht von dem Fall, dass mehrere Nachbarn senden unterscheiden. Die Kommunikation der hier vorgestellten Protokolle läuft synchronisiert in Zeitabschnitten ab.

Es wird ein randomisiertes Protokoll vorgestellt, dass für den Broadcast einer Nachricht in eben solchen Netzwerken, mit Wahrscheinlichkeit $1 - \epsilon$, $O((D + \log \frac{n}{\epsilon}) * \log n)$ Zeitabschnitte benötigt, wobei n die Anzahl der Knoten und D der Durchmesser des Netzwerks ist. Ferner wird bewiesen, dass für das Broadcasting von beliebigen deterministischen Protokollen eine untere Schranke $\Omega(n)$ für die Anzahl der benötigten Zeitabschnitte existiert. Diese Schranke gilt sogar für Netzwerke mit Durchmesser gleich 3. Diese beiden Ergebnisse zeigen einen exponentiellen Unterschied in der Laufzeit bei randomisierten und deterministischen Protokollen.

1 Einleitung

Um eine Nachricht in einem Netzwerk ausgehend von einem Teilnehmer an alle anderen zu verteilen (Broadcast), müssen bestimmte Besonderheiten beachtet werden, die bei der Kommunikation in Netzen auftreten. Ein Netzwerk lässt sich als ungerichteter Graph G modellieren (siehe Abbildung 1) wobei die Knoten den Netzwerkteilnehmern (im Folgenden auch Prozessoren genannt) entsprechen. Eine Kante zwischen zwei Knoten bedeutet, dass die beiden Knoten jeweils die gesendeten Nachrichten des Anderen empfangen können (in Abbildung 1 z.B. empfangen die Knoten 2 und 5 die Nachrichten von 1 und umgekehrt). An dem Beispiel ist leicht zu erkennen, dass Knoten 1 keine Nachrichten auf direktem Wege an z.B. Knoten

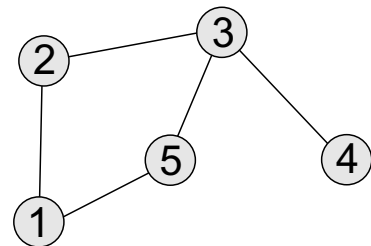


Abbildung 1: Ein einfaches Netzwerk als Graph

3 senden kann. Die Nachrichten müssten über Knoten 2 oder 5 an Knoten 3 weitergeleitet werden. Wenn der Graph einen Durchmesser $D > 1$ hat spricht man von einem Multi-Hop Netzwerk (ein Single-Hop Netzwerk würde einem vollständigen Graph mit $D = 1$ entsprechen).

Damit die Nachrichten vollständig und korrekt empfangen werden können wird davon ausgegangen, dass alle Prozessoren eines Netzwerks ihre Sende- und Empfangstätigkeit in synchronisierten Zeitabschnitten tätigen.

In einem Funknetzwerk teilen sich alle Teilnehmer dasselbe Medium (den Funkkanal). Problematisch wird es nun wenn mehrere Teilnehmer gleichzeitig versuchen eine Nachricht auf dem Medium zu senden. Wenn zum Beispiel Knoten 2 und 5 gleichzeitig senden, kommt es bei den Knoten 1 und 3 zur Kollision und die Nachricht wird nicht ausgeliefert. Gerade in Funknetzwerken ist es schwierig eine Kollision von dem ständig vorhandenen Rauschen auf dem Funkkanal zu unterscheiden.

Um die verschiedenen Probleme die bei Netzwerken entstehen in den Griff zu bekommen, werden Netze auch als Reihenfolge übereinander gestapelter Schichten dargestellt. Dabei ist jede Schicht für ein bestimmtes Problem zuständig und stellt seine Dienste der nächst höheren Schicht zur Verfügung. Das ISO-OSI Referenzmodell mit seinen 7 Schichten [Tanenbaum00] ist in diesem Umfeld sehr verbreitet. Dabei finden sich die im weiteren Verlauf dieser Arbeit besprochenen Protokolle in der 2. Schicht (Sicherheitsschicht) des Modells wieder. Genauer gesagt beschäftigen sie sich mit dem Problem, welches entsteht wenn mehrere Teilnehmer auf ein gemeinsames Medium zugreifen wollen. Für diese Aufgabe ist eine spezielle Zwischenschicht der Sicherheitsschicht zuständig, die MAC¹-Teilschicht.

Wenn im weiteren Verlauf ein vorgestellter Algorithmus als Broadcast Protokoll bezeichnet wird, so entspricht er folgender Definition.

Definition 1. : *(Broadcast Protokolle) Ein Broadcast Protokoll für Funknetzwerke ist ein Multiprozessor Protokoll, das in Zeitabschnitten $(0,1,\dots)$ folgendermaßen abläuft:*

1. *Im ersten Zeitabschnitt ($t=0$) sendet ein bestimmter Prozessor, die Quelle, eine Nachricht.*
2. *In jedem Zeitabschnitt inkl. $t=0$, agiert jeder Prozessor entweder als Sender, Empfänger oder ist nicht aktiv.*
3. *Ein Prozessor enthält die Nachricht im Zeitabschnitt t , genau dann wenn er in diesem Zeitabschnitt als Empfänger agiert und genau ein Nachbar als Sender agiert.*
4. *Ein Prozessor kann nur als Sender agieren, wenn er in einem vorigen Zeitabschnitt eine Nachricht empfangen hat.*
5. *Der Broadcast ist vollständig im Zeitabschnitt t , wenn alle Prozessoren die Nachricht in einem der Zeitabschnitte $0,\dots,t$ erhalten haben.*

Die Netzwerke auf denen die vorgestellten Algorithmen arbeiten sind Multi-Hop Netzwerke deren Topologie vorher nicht bekannt ist. In Abschnitt 2 wird eine obere Schranke für einen randomisierten Broadcast Algorithmus vorgestellt und in Abschnitt 3 wird eine untere Schranke für beliebige deterministische Broadcast Algorithmen gezeigt.

¹Medium Access Control

2 Obere Schranke

2.1 Der Decay-Algorithmus

Die Grundlage für die im weiteren Verlauf dieses Abschnitts vorgestellten randomisierten Protokolle bildet der Decay-Algorithmus (siehe Abbildung 2). Dieser Algorithmus garantiert dass eine Nachricht einen Knoten v , ausgehend von d informierten Nachbarn, mit Wahrscheinlichkeit von mindestens $\frac{1}{2}$ erreicht. Um eine Weiterleitung der Nachricht zu erreichen, muss nach Definition des zugrunde liegenden Modells genau einer der d informierten Nachbarn senden. Für $d \geq 2$ müssen nun genau $d - 1$ Nachbarn aufhören zu senden. Da die Topologie des Netzwerks aber nicht bekannt ist, hat ein Nachbar auch keine Informationen über die anderen Nachbarn und vor allem nicht, welche dieser Nachbarn bereits informiert sind. Somit muss ein Knoten selbst entscheiden, ob er aufhört zu senden oder nicht, und zwar ohne Kenntnis über das Netzwerk. Diese Entscheidung trifft ein Knoten durch das Werfen einer Münze, somit wird erreicht, dass der Knoten in jeder Runde mit einer Wahrscheinlichkeit von $\frac{1}{2}$ aufhört zu senden. Um im Fall $d = 1$ eine Weiterleitung der Nachricht zu garantieren sendet ein Knoten zuerst die Nachricht und entscheidet dann ob er aufhört oder nicht.

```
Decay( $k, m$ )
  repeat höchstens  $k$  mal ( $k \geq 1$ )
    Sende Nachricht  $m$  an alle Nachbarn
    Wähle  $c \in \{0, 1\}$  zufällig und gleichverteilt
  until  $c = 0$ 
end
```

Abbildung 2: Der Decay Algorithmus

Lemma 1. *Sei y ein Knoten aus G und $d \geq 2$ Nachbarn von y führen den Decay-Algorithmus aus. $P(k, d)$ wird definiert als die Wahrscheinlichkeit, dass y die Nachricht m innerhalb von k Runden erhält, bei d informierten Nachbarn. Es gilt ($\log = \text{Logarithmus zur Basis } 2$):*

$$(i) \lim_{k \rightarrow \infty} P(k, d) \geq \frac{2}{3}; \quad (ii) \text{ für } k \geq 2 \log d, P(k, d) \geq \frac{1}{2};$$

Beweis: zu (i):

Da $0 \leq P(k, d) \leq 1$ und $P(0, d) = 0$ gilt, und $P(k, d)$ für festes d mit k monoton steigt (Beweis durch Induktion über k , intuitiv: je mehr Runden zur Verfügung stehen, desto höher ist die Wahrscheinlichkeit, dass die Nachricht an y weitergeleitet wird). Daraus folgt, dass $\lim_{k \rightarrow \infty} P(k, d)$ existiert. Wir schreiben daher $P(\infty, d) := \lim_{k \rightarrow \infty} P(k, d)$

Offensichtlich gilt:

$P(k, 0) = 0$ [da hier kein Nachbar informiert ist]

$P(1, 1) = 1$ [da der einzige informierte Nachbar in der ersten Runde sendet]

$P(1, d) = 0$ für $d \geq 2$ [da in der ersten Runde alle d Nachbarn senden und es zu einer Kollision kommt.]

Wir betrachten nun den Fall $P(2, 2)$, d.h. es sind 2 Nachbarn informiert und es stehen 2 Runden zur Verfügung. Es können die folgenden Fälle auftreten.

Fall 1: 2 Nachbarn hören nach der ersten Runde auf zu senden. Dies passiert mit Wahrschein-

lichkeit $\frac{1}{4}$ und in der noch verbleibenden Runde erhalten wir die Wahrscheinlichkeit $P(1,0) = 0$

Fall 2: 1 Nachbar hört auf zu senden. Die Wahrscheinlichkeit für diesem Fall beträgt $2 * \frac{1}{4} = \frac{1}{2}$ und in Runde 2 gilt dann $P(1,1) = 1$.

Fall 3: 0 Nachbarn hören auf zu senden, was mit Wahrscheinlichkeit $\frac{1}{4}$ passiert und für die verbleibende Runde haben wir $P(1,2) = 0$.

Dann erhalten wir:

$$P(2,2) = \overbrace{\frac{1}{4} \cdot P(1,0)}^{\text{Fall 1}} + \overbrace{\frac{1}{2} \cdot P(1,1)}^{\text{Fall 2}} + \overbrace{\frac{1}{4} \cdot P(1,2)}^{\text{Fall 3}} = \frac{1}{4}(1 \cdot \overbrace{P(1,0)}^0) + 2 \cdot \overbrace{P(1,1)}^1 + 1 \cdot \overbrace{P(1,2)}^0 = \frac{1}{2}$$

Für $P(k,d)$ mit $d \geq 2$ lässt sich aus diesem Ergebnis die folgende Formel ableiten

$$\begin{aligned} P(k,d) &= \sum_{i=0}^d P(i \text{ Nachbarn hören auf}) \cdot P(k-1, d-i) = \sum_{i=0}^d \binom{d}{i} \frac{1}{2}^i \cdot \frac{1}{2}^{d-i} \cdot P(k-1, d-i) \\ &= \sum_{i=0}^d \binom{d}{i} 2^{-d} P(k-1, d-i) = 2^{-d} \sum_{i=0}^d \binom{d}{i} P(k-1, i) \end{aligned}$$

Mit Hilfe dieser Rekursion können wir nun (i) beweisen. Da $P(\infty,0) = 0$ und $P(\infty,1) = 1$ gilt für $d \geq 2$:

$$P(\infty, d) = 2^{-d} \sum_{i=0}^d \binom{d}{i} P(\infty, i) = 2^{-d} \sum_{i=1}^d \binom{d}{i} P(\infty, i)$$

Beweis durch Induktion über d ($d \geq 2$)

Induktionsstart: mit $d = 2$

$$P(\infty, 2) = 2^{-2} \sum_{i=1}^2 \binom{2}{i} P(\infty, i) = \frac{1}{4} \left(\binom{2}{1} P(\infty, 1) + \binom{2}{2} P(\infty, 2) \right) = \frac{1}{2} + \frac{1}{4} \cdot P(\infty, 2) = \frac{2}{3}$$

Induktionsschritt: sei $d > 2$. Unter der Annahme, dass $P(\infty, i) \geq \frac{2}{3}$ für alle $i \leq d$ gilt erhalten wir:

$$\begin{aligned} P(\infty, d) \cdot (2^d - 1) &= \sum_{i=1}^{d-1} \binom{d}{i} P(\infty, i) = \binom{d}{1} \underbrace{P(\infty, 1)}_1 + \sum_{i=2}^{d-1} \binom{d}{i} P(\infty, i) \\ &= d + \sum_{i=2}^{d-1} \binom{d}{i} \underbrace{P(\infty, i)}_{\geq \frac{2}{3}} \geq d + \frac{2}{3} \sum_{i=2}^{d-1} \binom{d}{i} > \frac{2}{3} (2^d - 1) \\ \Rightarrow P(\infty, d) &> \frac{2}{3} \end{aligned}$$

□

zu (ii):

Für $d \leq 5$ setzen wir die Werte einfach in die Rekursionsgleichung $P(k, d) = 2^{-d} \sum_{i=1}^d \binom{d}{i} P(k-1, i)$ aus dem ersten Teil des Beweises ein. Dann erhält man nebenstehende Tabelle und wie man sieht gilt $P(k, d) \geq \frac{1}{2}$ für $d \leq 5$ und $k \geq 2 \log d$. Die Einträge, die in Tabelle 2 mit (*) gekennzeichnet worden sind, weisen auf unzulässige Werte hin, da dort $k < 2 \log d$ ist

$d \setminus k$	2	3	4	5
2	$\frac{1}{2}$	$\frac{5}{8}$	$\frac{21}{32}$	$\frac{85}{128}$
3	(*)	$\frac{9}{16}$	$\frac{87}{128}$	$\frac{723}{1024}$
4	(*)	$\frac{7}{16}$	$\frac{167}{256}$	$\frac{2895}{4096}$
5	(*)	(*)	(*)	$\frac{313}{512}$

Tabelle 2: Werte für $P(k, d)$

Betrachte den Fall $d \geq 6$:

Sei $T_{t,d}$ eine Indikatorzufallsvariable² für den Fall „Alle Nachbarn von y haben zum Zeitpunkt t aufgehört zu senden“. Sei R_d eine Indikatorzufallsvariable für den Fall „ y erhält die Nachricht in endlicher Zeit“. Wenn beide Ereignisse eingetreten sind, bedeutet dies, dass y die Nachricht in endlicher Zeit t erhält. Also ist die Wahrscheinlichkeit, dass y die Nachricht in endlicher Zeit k erhält ($P(k, d)$) größer gleich der Wahrscheinlichkeit, dass die beiden oben genannten Ereignisse eintreten.

$$\begin{aligned}
 P(k, d) &\geq Pr(R_d \wedge T_{k,d}) = 1 - Pr(\overline{R_d}) - Pr(\overline{T_{k,d}}) \\
 &\geq \underbrace{P(\infty, d)}_{\text{siehe (i)}} - \underbrace{d \cdot 2^{-k}}_{k \geq 2 \log d} \geq \frac{2}{3} - d \cdot d^{-2} \geq \frac{1}{2} \text{ für } d \geq 6
 \end{aligned}$$

2.2 Das Broadcast Protokoll

```

procedure Broadcast;
   $k := 2 \lceil \log \Delta \rceil$ ;
   $t := 2 \lceil \log \frac{N}{\epsilon} \rceil$ ;
  Warte bis eine Nachricht (m) empfangen wurde;
  Wiederhole  $t$  mal
    Warte bis  $(\text{Zeit} \bmod k) = 0$ ;
    Decay( $k, m$ );
end
  
```

Abbildung 3: Die Broadcast Prozedur

Die Broadcast Prozedur aus Abbildung 3 ruft den Decay-Algorithmus mehrere Male auf um zu erreichen, dass eine Nachricht auch wirklich mit einer Wahrscheinlichkeit von mindestens $\frac{1}{2}$ weitergeleitet wird. Sollte der Parameter k mindestens $2 \log d$ sein (siehe Teil (ii) von Lemma 1). Da der Parameter d , also die Anzahl der informierten Nachbarn nicht bekannt ist wird hier eine obere Schranke Δ angegeben, die den Eingrad³ nach oben beschränkt. Da Lemma 1 fordert, dass die Ausführungen des Decay-Algorithmus synchronisiert ablaufen, werden sie hier nur zu Zeiten gestartet, die einem Vielfachen von k ($\text{Zeit} \bmod k$) entsprechen. Hierbei wird davon ausgegangen das die Zeit jedem Netzwerkteilnehmer bekannt ist.

²Variable die den Wert 1 annimmt wenn das Ereignis eintritt und sonst den Wert 0 hat

³der Eingrad eines Knotens ist die Anzahl der eingehenden Kanten

Bei der Ausführung des randomisierten Broadcast Protokolls führt jeder Teilnehmer des Netzwerks die Broadcast-Prozedur aus Abbildung 3 aus und es gibt eine Quelle s , von der ausgehend die Nachricht verbreitet wird.

Die Frage die sich nun stellt ist, in welcher Zeit und mit welcher Wahrscheinlichkeit die Nachricht jeden Knoten erreicht. Zunächst einmal wird die Frage betrachtet mit welcher Wahrscheinlichkeit das randomisierte Broadcast Protokoll erfolgreich ist, also alle Knoten informiert.

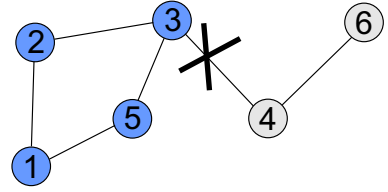


Abbildung 4: Nachricht „hängt“ bei Knoten 3 (Nicht informierte Knoten sind grau, informierte blau)

Lemma 2. *Wenn ein Netzwerk das randomisierte Broadcast-Protokoll ausführt, gilt:*

$$Pr(\text{Alle Knoten erhalten } m) \geq 1 - \epsilon$$

Beweis: Wenn eine Nachricht nicht alle Knoten erreicht, heisst das, dass es mindestens einen Knoten gibt, der nicht informiert wird. Das bedeutet, es gibt eine Stelle im Netzwerk an der die Nachricht „hängt“, d.h. es gibt einen Knoten der mindestens einen informierten Nachbarn hat, selbst aber nicht informiert ist (in Abbildung 4 gilt dies für Knoten 4). Sei X die Wahrscheinlichkeit für das Ereignis „Nicht alle Knoten erhalten m “

$$\begin{aligned} Pr(X) &= Pr(\exists v \neq s \text{ mit } v \text{ hat } m \text{ nicht erhalten und einer von } v\text{'s Nachbarn hat } m \text{ erhalten}) \\ &\leq \sum_{v \neq s} Pr(v \text{ hat } m \text{ nicht erhalten und einer von } v\text{'s Nachbarn hat } m \text{ erhalten}) \\ &\leq \sum_{v \neq s} Pr(v \text{ hat } m \text{ nicht erhalten} \mid \text{einer von } v\text{'s Nachbarn hat } m \text{ erhalten}) \\ &\leq n \cdot \left(\frac{1}{2}\right)^{\lceil \log \frac{n}{\epsilon} \rceil} \leq n \cdot \left(\frac{\epsilon}{n}\right) \leq \epsilon \end{aligned}$$

Lemma 2 sagt nur etwas über die Wahrscheinlichkeit aus mit der alle Knoten informiert werden, jedoch nicht zu welchem Zeitpunkt dies passiert. Eine offensichtliche Schranke stellt $O(Dk \cdot \log \frac{n}{\epsilon})$ dar, denn die Broadcast-Prozedur hat eine Laufzeit von $t \cdot k$ und wird von jedem Teilnehmer auf dem längsten Weg D (= Durchmesser) im Netzwerk ausgeführt. Das folgende Lemma stellt eine bessere Schranke für die Laufzeit des randomisierten Broadcast-Protokolls dar.

Lemma 3. *Sei $T(\epsilon) = 2D + 5\sqrt{\log \frac{n}{\epsilon}} \cdot \max(\sqrt{D}, \sqrt{\log \frac{n}{\epsilon}})$. Betrachte die Ausführung des randomisierten Broadcast-Protokolls mit $t = \infty$, dann gilt für alle $0 < \epsilon \leq 1$:*

(i) *Sei T_v eine Zufallsvariable für die Zeit in der Knoten v die Nachricht m erhält. Dann gilt $\forall v \in V$:*

$$Pr(T_v > 2\lceil \log \Delta \rceil \cdot T(\epsilon)) < \frac{\epsilon}{n}$$

(ii) *Sei $T_{fin} = \max_v T_v$, dann gilt:*

$$Pr(T_{fin} \leq 2\lceil \log \Delta \rceil \cdot T(\epsilon)) > 1 - \epsilon$$

Der Beweis zu diesem Lemma wird im Anhang B gegeben.

Wenn man nun Lemma 2 und 3 miteinander kombiniert erhält man:

Lemma 4. *Alle Knoten erhalten die Nachricht mit Wahrscheinlichkeit $\geq 1 - 2\epsilon$ in Zeit $2\lceil \log \Delta \rceil \cdot T(\epsilon)$*

3 Untere Schranke für deterministisches Broadcasting

Die untere Schranke wird gezeigt, indem das Broadcasting auf einer bestimmten Klasse von Graphen betrachtet wird und das Problem in mehreren Schritten auf ein kombinatorisches Spiel reduziert wird.

Die betrachtete Graphenklasse C_n ist wie folgt definiert. Sei $S \subseteq \{1, 2, \dots, n\}$ mit $S \neq \emptyset$. Für alle Graphen $G_S \in C_n$ mit $G_S = (V, E)$ und $V = \{0, 1, 2, \dots, n, n+1\}$, gilt:

$$E_1 = \{(0, i) : 1 \leq i \leq n\}$$

$$E_2 = \{(i, n+1) : i \in S\}$$

(siehe Abbildung 5)

Dabei wird der Knoten 0 als Quelle bezeichnet und Knoten $n+1$ als Senke. Man spricht hier auch von Ebenen. Die Quelle stellt Ebene 1 dar, die Senke Ebene 3 und die Knoten $p \in \{1, \dots, n\}$ bilden die 2. Ebene (siehe Abbildung 5). Ein Broadcast auf einem Graphen aus der Klasse C_n endet, wenn die von der Quelle gesendete Nachricht die Senke erreicht hat.

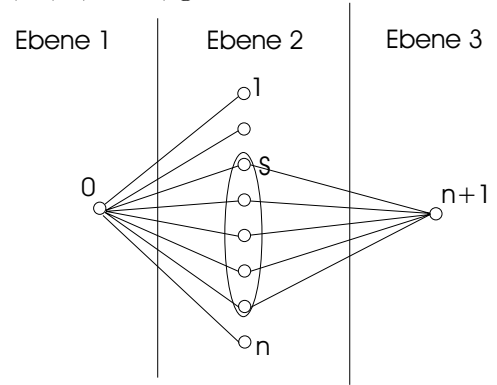


Abbildung 5: Die Graphenklasse C_n

3.1 Reduzierung des Problems

Die Reduktion erfolgt in drei Schritten. In den ersten beiden Schritten wird das Problem zunächst vereinfacht, um es dann im letzten Schritt als kombinatorisches Spiel, dem sogenannten *Hitting Game*, zu formulieren. Zunächst muss die in der Einleitung vorgestellte Definition von Broadcast Protokollen für deterministische Protokolle erweitert werden.

Definition 2. (*deterministische Broadcast Protokolle*) Ein deterministisches Broadcast Protokoll ist ein Broadcast Protokoll erweitert um folgende Eigenschaften

1. Die Aktion eines Prozessors in einem Zeitabschnitt wird bestimmt durch seine initiale Eingabe (bestehend aus eigener ID und den ID's der Nachbarn) und den Nachrichten, die er in vorhergehenden Zeitabschnitten empfangen hat.
2. Ein Prozessor kann die Nachricht in einem Zeitabschnitt ebenfalls erhalten, wenn er als Empfänger agiert und mehr als ein Nachbar als Sender agiert. In diesem Fall kann die Übermittlung der Nachricht allerdings nicht garantiert werden.

zu Punkt 1: Anhand des Beispielgraphen aus Abbildung 1 lässt sich schnell zeigen, dass es keinen deterministischen Algorithmus gibt, der diesen Graph ohne ID's informieren kann. Wenn alle Knoten den gleichen deterministischen Algorithmus ausführen, dann senden sie auch immer zur gleichen Zeit und es würde bei Knoten 3 immer zur Kollision kommen.

zu Punkt 2: Bisher wurde davon ausgegangen dass der Fall, dass kein Nachbar sendet nicht unterschieden werden kann von dem Fall, dass mehrere Nachbarn senden. Das bedeutet aber auch gleichzeitig dass der Fall, dass genau ein Nachbar sendet immer unterschieden werden kann von dem Fall, dass mehr als ein Nachbar sendet. Da man nicht davon ausgehen kann, dass dies immer möglich ist wird das Protokoll an dieser Stelle etwas „aufgelockert“. Eine detailliertere Begründung, warum diese Erweiterung nötig ist wird in [BDI03] gegeben.

Definition 3. (*eingeschränkte Broadcast Protokolle*) Ein deterministisches Broadcast Protokoll Π für die Graphenklasse C_n wird als eingeschränkt bezeichnet, wenn für jeden Graph

$G_S \in C_n$ und jeden Zeitabschnitt i gilt, dass im i -ten Zeitabschnitt entweder die Quelle oder die Senke, aber niemals beide gleichzeitig aktiv sind.

Lemma 5. *Wenn es ein deterministisches Broadcast Protokoll gibt, welches in t Zeitabschnitten auf jedem Netzwerk aus C_n terminiert, dann existiert ein eingeschränktes Broadcast Protokoll, das auf jedem Netzwerk aus C_n in $2t$ Zeitabschnitten terminiert.*

Der Beweis wird in [BDI92] im Appendix A1 gegeben.

Definition 4. *Der S -Indikator eines Prozessors aus der zweiten Ebene $p \in 1, 2, \dots, n$ bezeichnet als χ_p^S ist 1, wenn $p \in S$ und 0 sonst.*

Definition 5. *(abstrakte Broadcast Protokolle) Ein abstraktes Broadcast Protokoll ist ein Multiprozessor Protokoll, welches in Runden $(1, 2, 3, \dots)$ folgendermaßen abläuft.*

1. *In jeder Runde können nur Prozessoren aus der 2. Ebene als Sender agieren und entweder die Quelle oder die Senke (aber nicht beide gleichzeitig) agieren als Empfänger. Alle gesendeten Nachrichten bestehen nur aus der ID und dem S -Indikator des Senders (p, χ_p^S) .*
2. *Ein Prozessor (Quelle oder Senke) erhält die Nachricht in einer Runde garantiert, wenn er als Empfänger agiert und genau ein Nachbar als Sender agiert. Sollten mehrere Nachbarn in dieser Runde als Sender agieren, kann der Prozessor die Nachricht ebenfalls empfangen (in diesem Fall kann die Übermittlung der Nachricht allerdings nicht garantiert werden). Eine Runde ist erfolgreich, wenn der als Empfänger agierende Prozessor eine Nachricht erhalten hat.*
3. *Am Ende jeder Runde wissen alle Prozessoren der 2. Ebene ob diese Runde erfolgreich war, und wen dem so ist kennen sie den Inhalt der empfangenen Nachricht.*
4. *Die Aktion eines Prozessors in einer Runde wird bestimmt, durch seine initiale Eingabe (bestehend aus eigener ID und den ID's der Nachbarn) und den Nachrichten, die er in vorhergehenden Zeitabschnitten empfangen hat.*
5. *Der Broadcast ist vollständig sobald der S -Indikator in einer erfolgreichen Runde 1 ist (entspricht dem Fall, dass eine Nachricht von einem Prozessor aus S empfangen wurde).*

Lemma 6. *Wenn es ein eingeschränktes Broadcast Protokoll gibt, das innerhalb von t Zeitabschnitten auf jedem Netzwerk aus C_n terminiert, dann gibt es ein abstraktes Broadcast Protokoll, das auf jedem Netzwerk aus C_n innerhalb von t Runden terminiert.*

Der Beweis kann in [BDI92] im Appendix A2 nachgelesen werden.

3.2 Das n -th Hitting Game

Das n -th *Hitting Game* ist ein kombinatorisches Spiel, welches von 2 Parteien gespielt wird, dem Sucher und dem Schiedsrichter. Das Spiel wird auf einer nicht leeren Menge $S \subseteq 1, \dots, n$ gespielt, die nur dem Schiedsrichter bekannt ist. Der Sucher hat die Aufgaben ein Element aus S zu treffen. Das Spiel läuft in Schritten ab und jeder Schritt i besteht aus folgenden Aktionen:

1. Der Sucher wählt eine Menge M_i aus und zeigt sie dem Schiedsrichter

2. Der Schiedsrichter vergleicht M_i mit S . Es treten folgende Fälle auf

- (a) $|M_i \cap S| = 1$: der Schiedsrichter zeigt dem Sucher das Element $M_i \cap S$ und das Spiel ist beendet
- (b) $|M_i \cap \bar{S}| = 1$: der Schiedsrichter zeigt dem Sucher das Element $M_i \cap \bar{S}$ ohne das Spiel zu beenden
- (c) $|M_i \cap S| \neq 1$ und $|M_i \cap \bar{S}| \neq 1$: Der Schiedsrichter antwortet mit der leeren Menge \emptyset

Der Sucher *gewinnt* das Spiel in t Schritten, wenn das Spiel im t -ten Schritt beendet wurde und der Schiedsrichter dem Sucher ein Element aus S übergeben hat. Wenn der Sucher unabhängig von S das Spiel in t -Schritten gewinnt, dann spricht man von einer *t-Schritt Gewinner Strategie*.

Lemma 7. *Wenn es ein abstraktes Broadcast Protokoll gibt, das für alle Netzwerke aus C_n in t Runden terminiert, dann existiert eine $2t$ -Schritt Gewinner Strategie für das n -th Hitting Game.*

Der Beweis kann in [BDI03] in Kapitel 2.1 nachgelesen werden.

Die von dem Sucher in einer Runde ausgewählte Menge M_i entspricht den in einer Runde sendenden Prozessoren (vgl. Definition 5, Punkt 1). Wenn der Schiedsrichter eine Antwort gibt, dann hat genau ein Prozessor aus S oder genau ein Prozessor aus \bar{S} die Nachricht gesendet und wurde folglich von der Senke oder der Quelle empfangen. Wenn der Schiedsrichter also eine Antwort gibt war die Runde erfolgreich (vgl. Definition 5 Punkt 2). Die Antwort des Schiedsrichters entspricht Punkt 3. Die Prozessoren des Netzwerks wissen also welcher Prozessor in einer erfolgreichen Runde gesendet hat, da sie nach Definition den Inhalt der Nachricht kennen, welche wiederum nur aus Prozessor-ID und S-Indikator besteht. Die Aktionen des Suchers entsprechen also denen eines deterministischen Algorithmus, nämlich in jeder Runde eine Menge von Prozessoren auszuwählen, die die Nachricht senden. Der Schiedsrichter entscheidet anhand des zugrunde liegenden Protokolls (ein abstraktes Broadcast Protokoll) ob eine Runde erfolgreich war und wenn ja, ob das Spiel dadurch beendet wird (und damit der Broadcast vollständig ist). Im folgenden Abschnitt wird ein Algorithmus gezeigt der für jede Strategie des Suchers mit nicht mehr als $\frac{n}{2}$ Schritten die Menge S so „gemein“ wählt, dass der Sucher immer verliert.

3.3 Eine untere Schranke für das Hitting Game

Es lässt sich zeigen, dass es eine Menge S gibt, die jede Strategie des Suchers mit $\frac{n}{2}$ Schritten „schlägt“. Es lässt sich ebenfalls zeigen, dass es eine Menge S gibt für die der Schiedsrichter mit der leeren Menge \emptyset antwortet, falls $|M_i| > 1$ ist für alle $0 \leq i < \frac{n}{2}$. Daraus ergibt sich, dass der Sucher keine neuen Informationen aus den Antworten des Schiedsrichters erhält, da er entweder \emptyset oder M_i als Antwort erhält. Es genügt also eine Menge S zu finden, die alle vergesslichen Strategien des Suchers schlägt. Eine vergessliche Strategie bedeutet in diesem Fall, dass die Auswahl der M_i nicht von den Ergebnissen der vorangegangenen Runden abhängt. Es wird nun ein Algorithmus vorgestellt, der eine Menge S mit den gewünschten Eigenschaften generiert. Auf Basis der Sequenz der einzelnen Rundenmengen M_i wird die Anfangsmenge $S = 1, 2, \dots, n$ sukzessive reduziert bis folgende Bedingungen gelten. Für alle $1 \leq i \leq t$ gilt:

```

procedure findSet;
input:  $M_1, M_2, \dots, M_t$ ;
 $S \leftarrow 1, 2, \dots, n$ ;
while (es existiert ein  $i$ , so dass  $|M_i \cap S| = 1$ ) do begin
     $x \leftarrow M_i \cap S$ ;
     $S \leftarrow S - \{x\}$ ;
    while (es existiert ein  $j$ , so dass  $|M_j \cap S| = |M_j| - 1 > 0$ ) do begin
        Wähle (beliebiges)  $p \in M_j \cap S$ ;
         $S \leftarrow S - \{p\}$ ;
    end;
end;
output  $S$ ;

```

Abbildung 6: Der FindSet-Algorithmus

Bedingung 1: Wenn $|M_i| = 1$ ist, dann gilt $M_i \cap S = \emptyset$, bzw. $M_i \subseteq \bar{S}$

Bedingung 2: Wenn $|M_i| > 1$, dann gilt $|M_i \cap S| \neq 1$ und $|M_i \cap \bar{S}| \neq 1$

Das folgende Beispiel soll den Ablauf des FindSet-Algorithmus aus Abbildung 6 verdeutlichen:

Beispiel 1. Sei $n = 6$, $t = 3$, $M_1 = \{3\}$, $M_2 = \{3, 4, 5, 6\}$, und $M_3 = \{1, 2, 3\}$ und $S = \{1, 2, \dots, 6\}$

		$M_1 = \{3\}$	$M_2 = \{3, 4, 5, 6\}$	$M_3 = \{1, 2, 3\}$	S \bar{S}	Aktion
(1)	$M_i \cap S$	{3}	{3, 4, 5, 6}	{1, 2, 3}	{1, 2, 3, 4, 5, 6}	$S \leftarrow S - \{3\}$
	$M_i \cap \bar{S}$	\emptyset	\emptyset	\emptyset	\emptyset	
(2)	$M_i \cap S$	\emptyset	{4, 5, 6}	{1, 2}	{1, 2, 4, 5, 6}	$S \leftarrow S - \{2, 5\}$
	$M_i \cap \bar{S}$	{3}	{3}	{3}	{3}	
(3)	$M_i \cap S$	\emptyset	{4, 6}	{1}	{1, 4, 6}	$S \leftarrow S - \{1\}$
	$M_i \cap \bar{S}$	{3}	{3, 5}	{2, 3}	{2, 3, 5}	
(4)	$M_i \cap S$	\emptyset	{4, 6}	\emptyset	{4, 6}	<i>output</i> $S = \{4, 6\}$
	$M_i \cap \bar{S}$	{3}	{3, 5}	{1, 2, 3}	{1, 2, 3, 5}	

Tabelle 3: Beispielhafter Ablauf des FindSet Algorithmus

Der Algorithmus sucht zunächst alle Mengen M_i deren Schnitt mit S genau einem Element x entspricht, in diesem Beispiel ist das die Menge M_1 mit $M_1 \cap S = \{3\}$. Also wird das Element $\{3\}$ aus S entfernt (siehe Tabelle Zeile (1)). Dadurch wird garantiert, dass $|M_i \cap S| \neq 1$ gilt. Es muss allerdings auch $|M_i \cap \bar{S}| \neq 1$ (für $|M_i| > 1$) gelten. Um dies zu erreichen, wird von allen M_j mit $|M_j \cap \bar{S}| = 1$ (bzw. $|M_j \cap S| = |M_j| - 1 > 0$) ein beliebiges Element aus $M_j \cap \bar{S}$ entfernt. Im Beispiel entspricht dies den Mengen M_2 und M_3 aus deren Schnittmenge mit S jeweils ein beliebiges Element (in diesem Fall, 2 und 5) aus S entfernt wird (siehe Tabelle Zeile (2)). Abschließend wird noch das Element $\{1\}$ aus S entfernt, da $M_3 \cap S = \{1\}$. Der

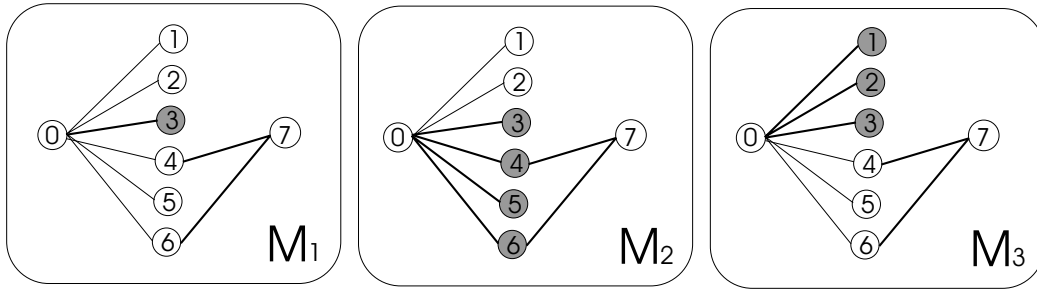


Abbildung 7: Übertragung des Beispiels auf einen Graphen

Algorithmus gibt die Menge $\{4, 6\}$ aus, für die die Bedingungen 1 und 2 gelten. Überträgt man dieses Ergebnis jetzt auf einen Graph (siehe Abbildung 7, die in einer Runde sendenden Prozessoren sind grau hinterlegt) sieht man schnell, dass die Senke nicht informiert wird.

Das der Algorithmus für das Beispiel korrekt arbeitet wurde bereits gezeigt. Jetzt gilt es noch zu zeigen, dass der Algorithmus immer die gewünschte Menge S ausgibt.

Lemma 8. Sei S die Ausgabe des FindSet-Algorithmus. Für alle $1 \leq i \leq t$ gilt:

- (1) $|M_i \cap S| \neq 1$
- (2) $|M_i \cap \bar{S}| = 1$ genau dann wenn $|M_i| = 1$

Beweis: Betrachte 2 Fälle:

Fall 1 ($M_i \cap S = \emptyset$):

Bedingung (1) gilt offensichtlich. In diesem Fall ist $M_i \subseteq \bar{S}$ und $|M_i \cap \bar{S}| = |M_i|$ und Bedingung (2) gilt ebenfalls.

Fall 2 ($M_i \cap S \neq \emptyset$):

Die äußere while-Schleife bewirkt, dass $|M_i \cap S| \neq 1$ ist und Bedingung (1) hält. Da $|M_i| \geq |M_i \cap S| > 1$ genügt es zu zeigen, dass $|M_i \cap \bar{S}| \neq 1$ ist. Dazu betrachten wir 2 Fälle:

$M_i = M_i \cap S$: In diesem Fall ist $M_i \cap \bar{S} = \emptyset$ und Bedingung (2) folgt.

$M_i \neq M_i \cap S$: In diesem Fall ist $1 \leq |M_i \cap \bar{S}| = |M_i - (M_i \cap S)|$. Durch die innere while-Schleife gilt $|M_i \cap S| \neq |M_i| - 1$ und dadurch $|M_i \cap \bar{S}| \neq 1$ □

Nun bleibt noch zu zeigen, dass der FindSet-Algorithmus auch ein nicht leeres S ausgibt.

Lemma 9. Wenn $t \leq \frac{n}{2}$ dann gibt der FindSet-Algorithmus $S \neq \emptyset$ aus.

Beweis: Man betrachte die Anzahl der Reduzierungen der Menge S . Man sieht leicht, dass jeder Schritt M_i zu maximal 2 Reduzierungen führen kann und zwar einmal durch die äußere while-Schleife und einmal durch die innere. Dadurch ergeben sich also maximal $2t$ Reduzierungen der Menge S . Eine etwas genauere Schranke lässt sich bestimmen indem man berücksichtigt, dass ein-elementige Schritte maximal zu einer Reduzierung von S führen können (durch die äußere while-Schleife) und die innere while-Schleife nur für Schritte mit $|M_i| > 1$ zu einer Reduzierung von S führen kann. Dadurch ergeben sich folgende Fälle:

Es ex. kein $|M_i| = 1$: Die Menge S wird nicht reduziert und der Algorithmus gibt $S = \{1, 2, \dots, n\}$ aus

Es ex. mindestens 1 $|M_i| = 1$: Die Menge S wird maximal $2(t - 1) + 1 \leq n - 1$ (für $t \leq \frac{n}{2}$) mal reduziert und $S \neq \emptyset$ folgt □

Nun können die Ergebnisse aus diesem Abschnitt zusammengefasst werden. Aus den Lemmata 5, 6, 7, 8 und 9 folgt:

Lemma 10. *Es existiert kein deterministischer Broadcast-Algorithmus, der in weniger als $\frac{n}{8}$ Zeitabschnitten terminiert*

Beweis: Für den Beweis müssen lediglich die Aussagen der Lemmata dieses Abschnitts zusammengefasst werden.

- Sei $T(n)$ die Anzahl der Zeitabschnitte bis das deterministische Broadcast Protokoll terminiert
- Sei $B(n)$ die Anzahl der Zeitabschnitte bis das eingeschränkte Broadcast Protokoll terminiert
- Sei $A(n)$ die Anzahl der Zeitabschnitte bis das abstrakte Broadcast Protokoll terminiert
- Sei $G(n)$ die Anzahl der Schritte die benötigt wird um das n -th Hitting Game zu gewinnen

$$T(n) \geq \overbrace{\frac{1}{2} \cdot B(n)}^{\text{Lemma 5}} \geq \overbrace{\frac{1}{2} \cdot A(n)}^{\text{Lemma 6}} \geq \overbrace{\frac{1}{4} \cdot G(n)}^{\text{Lemma 7}} \geq \overbrace{\frac{1}{4} \cdot \frac{n}{2}}^{\text{Lemma 8, 9}} = \frac{n}{8}$$

□

Es wurde gezeigt, dass Netzwerke existieren auf denen eine deterministische Zeitschranke von $\Omega(n)$ für das Broadcasting-Problem existiert, wohingegen es einen randomisierten Algorithmus gibt, der das Problem in $O(\log n)$ Zeit löst.

4 Anhang

A Chernoff-Schranke

Sei Y_i eine unabhängige Zufallsvariable mit $Y_i \in \{0, 1\}$ für $\epsilon \leq 1$ und $Y = \sum_{i=1}^k Y_i$, dann gilt:

$$Pr\left(Y < (1 - \epsilon) \cdot E[Y]\right) < e^{-\frac{1}{2}\epsilon^2 \cdot E[Y]}$$

B Beweis zu Lemma 3

Beweis: zu (i) Sei $Dist_i$ eine Zufallsvariable, die die Länge des kürzesten Weges von der Menge der in Runde i informierten Knoten (diejenigen, die Nachricht m bereits erhalten haben) zu Knoten v beschreibt. Eine Runde bezeichnet in diesem Fall eine der t synchronisierten Ausführungen des Decay-Algorithmus in dem randomisierten Broadcast Protokoll. Daraus folgt, dass eine Runde $2\lceil \log \Delta \rceil$ Zeitabschnitte benötigt. Da zum Zeitpunkt $t = 0$ nur die Quelle informiert ist, gilt $Dist_0 \leq D$.

Lemma 1 (ii) besagt, dass die Wahrscheinlichkeit, dass ein Knoten von seinen (bereits informierten) Nachbarn die Nachricht erhält $\geq \frac{1}{2}$ ist. Also ist die Wahrscheinlichkeit, dass der

$Dist_i$ innerhalb einer Runde um 1 verkürzt wird ebenfalls $\geq \frac{1}{2}$ (unter der Voraussetzung, dass Knoten v nicht bereits informiert ist also $Dist_i \neq 0$ ist).

$$Pr(Dist_i - Dist_{i+1} = 1 \mid Dist_i \neq 0) \geq \frac{1}{2}$$

Gesucht ist die Wahrscheinlichkeit, dass ein Knoten Nachricht m nicht innerhalb von $T(\epsilon) \cdot 2\lceil \log \Delta \rceil$ Zeitabschnitten erhält, also $Pr(Dist_{T(\epsilon)} > 0)$.

$$Pr(Dist_{T(\epsilon)} > 0) = Pr\left(\sum_{i=0}^{T(\epsilon)-1} (Dist_i - Dist_{i+1}) < Dist_0\right) \leq Pr\left(\sum_{i=0}^{T(\epsilon)-1} (Dist_i - Dist_{i+1}) < D\right)$$

Um diese Wahrscheinlichkeit mit Hilfe der Chernoff-Schranke (Anhang A) berechnen zu können wird zunächst die Zufallsvariable $\chi_i = Dist_i - Dist_{i+1}$ eingeführt. Man beachte, dass $Pr(\chi_i = 1) \geq \frac{1}{2}$ und damit $E[\sum_{i=0}^{T(\epsilon)-1} \chi_i] \geq \frac{T(\epsilon)}{2}$ ist. Weiterhin wird $D = (1 - \epsilon) \cdot E[\sum_{i=1}^k Y_i]$ gesetzt, daraus folgt $\epsilon = 1 - \frac{2D}{T(\epsilon)}$. Zur weiteren Vereinfachung sei $M(\epsilon) = \sqrt{\log \frac{n}{\epsilon}}$

$$\begin{aligned} Pr\left(\sum_{i=0}^{T(\epsilon)-1} \chi_i < D\right) &< e^{-\frac{1}{2}\epsilon^2 \cdot E[\sum_{i=0}^{T(\epsilon)-1} \chi_i]} \leq e^{-(1 - \frac{2D}{T(\epsilon)})^2 \cdot \frac{T(\epsilon)}{4}} \\ &\leq e^{-\left(\frac{5M(\epsilon) \cdot \max\{\sqrt{D}, M(\epsilon)\}}{2D + 5M(\epsilon) \cdot \max\{\sqrt{D}, M(\epsilon)\}}\right)^2 \cdot \frac{2D + 5M(\epsilon) \cdot \max\{\sqrt{D}, M(\epsilon)\}}{4}} \\ &\leq e^{-\frac{25M(\epsilon)}{4} \cdot \frac{\max\{D, M(\epsilon)^2\}}{2D + 5M(\epsilon) \cdot \max\{\sqrt{D}, M(\epsilon)\}}} \\ &\leq e^{-\frac{25M(\epsilon)}{4} \cdot \frac{1}{2+5}} \leq 2^{-M(\epsilon)^2} = 2^{-\log \frac{n}{\epsilon}} = \frac{\epsilon}{n} \end{aligned}$$

□

zu (ii):

$$Pr(T_{fin} \leq 2\lceil \log \Delta \rceil \cdot T(\epsilon)) = 1 - Pr(T_v > 2\lceil \log \Delta \rceil \cdot T(\epsilon)) \text{ und (ii) folgt.}$$

Literatur

- [BDI92] Bar-Yehuda, R., Goldreich, O., Itai, A.: *On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap Between Determinism and Randomization*, JCSS, 1992
- [BDI03] Bar-Yehuda, R., Goldreich, O., Itai, A.: *Errata Regarding (On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap Between Determinism and Randomization)*
- [Tanenbaum00] Tanenbaum, A.: *Computernetzwerke*, Pearson Studium, München, 2000