Hochschule Mittweida
Mathematisches Seminar

in der Fachrichtung
Diskrete und computerorientierte Mathematik

Wintersemester 2004/2005

Thema:

Die Komplexität des Zuverlässigkeitsproblems in verteilten Systemen

Erarbeitet von: Robert Siegfried
Weitzelstraße 15
09648 Mittweida
Tel.: (0 37 27) 99 92 17
E-Mail: rs@rsiegfried.de

Dozent: Prof. Dr. Klaus Dohmen

Abgabedatum: 20. Januar 2005

Inhaltsverzeichnis

1 Einleitung und Problemstellung

Unter einem verteilten System (engl. Distributed Computing System, DCS) wird eine Menge unabhängiger Computer (engl. processing elements) verstanden, welche dem Benutzer wie ein einzelnes, kohärentes System erscheint. Dies bedingt, dass die Computer über geeignete Kommunikationsverbindungen untereinander verfügen müssen, um benötigte Daten- oder Programmelemente austauschen zu können.[9]

Verteilte Systeme bieten potenziell eine Erhöhung der Zuverlässigkeit, des Durchsatzes, der Fehlertoleranz, der Ressourcenteilung und der Erweiterbarkeit. Während der Performancezuwachs durch parallel laufende Programme erreicht wird, ergibt sich die erhöhte Zuverlässigkeit und Fehlertoleranz durch die Redundanz der zu verarbeitenden Programme und Daten, sowie durch die Redundanz der Kommunikationsverbindungen.[4]

Insbesondere ist bei verteilten Systemen die erhöhte Zuverlässigkeit von großer Wichtigkeit. In diesem Kontext sind diverse Zuverlässigkeitsmaße entwickelt worden, um die Zuverlässigkeit von verteilten Systemen zu beschreiben. Ein zentraler Punkt bei der Entwicklung dieser Zuverlässigkeitsmaße ist, dass die Folgen aus der redundanten Verteilung von Daten und Programmen korrekt erfasst werden.[4, 5]

Diese Seminararbeit wird sich mit der algorithmischen Komplexität der Berechnung eines speziellen Zuverlässigkeitsmaßes, der *Distributed Program Reliability*, befassen. Hierzu werden zuerst die grundlegenden Begriffe, sowie die verwendete Symbolik definiert (Kap. 2). Im darauf folgenden Hauptteil dieser Seminararbeit wird gezeigt, dass die Berechnung der Distributed Program Reliability NP-hart ist und diese Aussage auch für bestimmte eingeschränkte Klassen von verteilten Systemen gültig ist (Kap. 3). Zum Schluss erfolgt eine kurze Zusammenfassung der Ergebnisse (Kap. 4).

2 Grundlagen

2.1 Distributed Program Reliability

Ein Distributed Computing System (DCS) besteht aus den folgenden Komponenten:

- Verarbeitungselemente (Knoten)
- Kommunikationsverbindungen (Kanten)
- Speicher
- Daten/Dateien
- Programme

Ein wichtiger Punkt bei der Konstruktion eines DCS ist die Zuverlässigkeit. Hierzu wurden verschiedene Maße eingeführt, wie beispielsweise die sog. *K-Terminal Reliability* (KTR). Die KTR beschreibt die Wahrscheinlichkeit, mit der eine bestimmte Menge von Knoten, welche eine Teilmenge aller Knoten ist, verbunden bleibt, wobei alle Kanten mit einer bestimmten Wahrscheinlichkeit statistisch unabhängig voneinander ausfallen können.

Hierbei ist zu beachten, dass die KTR wenig aussagekräftig ist, da die redundante Verteilung von Daten und Programmen nicht beachtet wird. Betrachtet man nun ein DCS in welchem die Knoten eine perfekte Zuverlässigkeit haben und die Kanten statistisch unabhängig voneinander mit bestimmten, bekannten Wahrscheinlichkeiten ausfallen, so lässt sich die *Distributed Program Reliability* (DPR) wie folgt definieren: Die DPR ist die Wahrscheinlichkeit, dass ein Programm mit verteilten Dateien erfolgreich verarbeitet wird unter der Annahme, dass Kanten ausfallen können. Die DPR beschreibt somit die Wahrscheinlichkeit, dass der Knoten, welcher das Programm enthält, sowie die Knoten, welche die benötigten Dateien enthalten und die benötigten verbindenden Kanten funktionsfähig sind. Im Unterschied zur KTR wird bei der DPR also nicht eine Menge von Knoten vorgegeben, welche verbunden bleiben müsen, sondern eine Menge von Dateien, welche für die korrekte Abarbeitung eines Programms erforderlich sind. Diese Abstraktion von (vorher definierten) bestimmten Knoten (auf denen eine Datei zur Verfügung steht) trägt dem Umstand Rechnung, dass in einem DCS Dateien redundant auf verschiedenen Knoten vorhanden sein können.[5, 4]

In dieser Arbeit werden durchgängig die folgenden Notationen verwendet:

$D = (V, E, F)$	Ungerichteter DCS-Graph mit Knotenmenge V, Kantenmenge E und der Menge F an Dateien, welche in D verteilt sind.
V	Menge der Knoten, alle mit perfekter Zuverlässigkeit.
E	Menge der Kanten, welche statistisch unabhängig voneinander mit bekannter Wahrscheinlichkeit ausfallen können.
F	Dateien, welche in D verteilt vorhanden sind. Hierbei erfolgt keine Unterscheidung zwischen Programmen und Daten.
$H \subseteq F$	Definierte Menge von Dateien, welche miteinander kommunizieren können müssen.
$FA_i \subseteq F$	Dateien, welche am Knoten i vorhanden sind.
p_i	Zuverlässigkeit von Kante i.
$R(D_H)$	DPR von D bei gegebenem H.

2.2 Komplexitätstheorie

Die Komplexitätstheorie untersucht die Komplexität von Funktionen und Algorithmen, d. h. sie beschäftigt sich mit dem für Berechnungen notwendigen Aufwand an Zeit oder Speicherplatz und insbesondere mit der Einteilung von Problemen in bestimmte Komplexitätsklassen.[6] Hierbei sind die Klassen $\mathcal{P}$ und $\mathcal{NP}$ von besonderer Bedeutung. Die Klasse $\mathcal{P}$ beinhaltet die Probleme, welche durch eine deterministische Turingmaschine in polynomialer Laufzeit gelöst werden können. Die Klasse $\mathcal{NP}$ hingegen beinhaltet diejenigen Probleme, welche durch eine nichtdeterministische Turingmaschine in Polynomialzeit gelöst werden können. Von besonderem Interesse sind hierbei die NP-vollständigen Probleme, da diese unter der (starken, aber nicht bewiesenen) Annahme, dass $\mathcal{P} \neq \mathcal{NP}$ gilt, nicht in polynomialer Zeit lösbar sind.

Um zu zeigen, dass ein Problem π NP-vollständig ist, geht man wie folgt vor:

1. Zeige, dass π in $\mathcal{NP}$ liegt.

2. Wähle ein bekanntes NP-vollständiges Problem π' aus.
3. Zeige, dass π' mittels einer Transformation f in Polynomialzeit auf π reduzierbar ist.

Ein Problem π, welches so schwierig ist, dass wir zwar zeigen können, dass π NP-vollständig ist (d. h. wir können eine geeignete Transformation f angeben), nicht aber, dass π in $\mathcal{NP}$ liegt, heißt *NP-hart*.[2, Kap. 2, 3, 5][3, Kap. 10]

3 Distributed Program Reliability

Im Folgenden wird gezeigt, dass die Berechnung der DPR sowohl für allgemeine DCS, als auch für eingeschränkte Klassen von DCS NP-hart ist. Soweit nicht anders vermerkt, basieren die Ausführungen in diesem Kapitel auf [5].

3.1 Allgemein

Satz 1 *Die Berechnung der DPR für ein allgemeines DCS ist NP-hart.*

Bevor wir den eigentlichen Beweis führen, wollen wir uns das Problem der Berechnung der K-Terminal Reliability (KTR) noch einmal anschauen. Die Fragestellung dieses Problems lautet wie folgt: Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit der Knotenmenge V und der Kantenmenge E. Jede Kante $e \in E$ kann mit einer bestimmten Wahrscheinlichkeit ausfallen, wobei diese unabhängig von den anderen Kanten ist. Des Weiteren ist eine Menge $K \subseteq V, |K| \geq 2$ gegeben. Gesucht ist die Größe $R(G_K)$, d. h. die Wahrscheinlichkeit, dass die Knotenmenge K in G zusammenhängend ist. Es lässt sich zeigen, dass dieses Problem NP-hart ist.[8, 10]

Beweis Um Satz 1 zu beweisen, werden wir das KTR-Problem auf das DPR-Problem reduzieren. Zu einem gegebenen Netzwerk $G = (V, E)$, sowie einer Menge $K \subseteq V$ können wir eine Instanz des DPR-Problems definieren. Hierzu konstruieren wir einen DCS-Graph $D = (V, E, F)$ mit derselben Topologie und denselben Zuverlässigkeiten der Kanten wie in G. Des Weiteren sei

$$FA_i = \begin{cases} \{f_i\} & \text{falls } i \in K \\ \emptyset & \text{sonst} \end{cases}$$

$$F = \bigcup_{\forall i \in K} \{f_i\}$$

Indem wir $H = F$ setzen, erhalten wir $R(D_H) = R(G_K)$. Da die Berechnung von $R(G_K)$ NP-hart ist, ergibt sich, dass auch die Berechnung der DPR NP-hart ist. $\square$

Als Konsequenz aus Satz 1 folgt, dass es äußerst unwahrscheinlich ist, dass ein effizienter Algorithmus zur Bestimmung der DPR existiert (vgl. Kap. 2.2). Eine Möglichkeit, die Komplexität der Berechnung zu verringern, besteht darin, die DPR nicht auf allgemeinen

Graphen zu definieren, sondern auf bestimmte Topologien einzuschränken. Im Folgenden sollen daher DCSse mit planaren Strukturen, Stern- und Baumtopologien, sowie 2-Baum- und serien-parallelen Topologien untersucht werden.

Da für derart eingeschränkte Klassen des KTR-Problems Polynomialzeit-Algorithmen entwickelt werden konnten, steht zu vermuten, dass dies auch für die Berechnung der DPR möglich ist. Wie im Folgenden aber gezeigt wird, ist dem nicht so.

3.2 Planare DCS

Satz 2 *Die Berechnung der DPR für ein planares DCS ist NP-hart.*

Beweis Aus dem Beweis von Satz 1 ist ersichtlich, dass das KTR-Problem lediglich ein Spezialfall des DPR-Problems ist. Da die Berechnung der KTR in planaren Graphen erwiesenermaßen NP-hart ist [5], folgt hieraus automatisch, dass auch die Berechnung der DPR in planaren Graphen NP-hart ist. □

3.3 DCS mit Stern-Topologie

Satz 3 *Die Berechnung der DPR für ein DCS mit Sterntopologie ist NP-hart. Dies gilt auch, falls $|FA_i| = 2$, d. h. falls jeder Knoten lediglich zwei Dateien bereithält.*

Ähnlich wie bei Satz 1 werden wir diesen Satz beweisen, indem wir ein bekanntes Problem auf das neue Problem reduzieren. Hierzu werden wir diesmal „Number of Edge Covers“ (#EC) verwenden. Dieses Problem lautet wie folgt: Gegeben sei ein ungerichteter Graph $G = (V, E)$. Eine Menge $L \subseteq E$ heißt Edge Cover, falls jeder Knoten von G ein Endknoten einer Kante aus L ist. Gesucht ist die Anzahl aller gültigen Edge Covers. Dieses Problem ist erwiesenermaßen NP-hart.[1]

Beweis Wie bereits angedeutet, reduzieren wir #EC auf unser Problem. Zu einem gegebenen Netzwerk $G = (V_1, E_1)$ mit $E_1 = \{e_1, e_2, \ldots, e_n\}$ konstruieren wir ein DCS $D = (V_2, E_2, F)$ mit einer Sterntopologie. Hierbei setzen wir $V_2 = \{s, v_1, v_2, \ldots, v_n\}$, d. h. V_2 besteht aus dem zentralen Knoten s des Sterns, sowie $n = |E_1|$ weiteren Knoten. Weiterhin sei $E_2 = \{(s, v_i) | 1 \leq i \leq n\}$ und $F = \{f_i | \forall i \in V_1\}$.

Des Weiteren sei $FA_{v_i} = \{f_u, f_v | e_i = (u, v) \in E_1\}$ für $1 \leq i \leq n$, $FA_s = \emptyset$ und $H = F$. In dem so konstruierten DCS definieren wir nun einen *datei-aufspannenden Baum* (FST, engl. file spanning tree, vgl. [4]) als einen Baum, dessen Knoten alle Dateien aus H bereithalten, d. h. $H \subseteq \bigcup_{\forall \text{Knoten } i \in \text{FST}} \{FA_i\}$. Aus der Konstruktion von D ist nun ersichtlich, dass es eine Eins-zu-Eins Korrespondenz zwischen den Edge Covers und den FSTs gibt.

Die DPR von D kann daher wie folgt ausgedrückt werden:

$$R(D_H) = \sum_{\forall \text{FST } t \in D} \left(\prod_{\forall \text{Kanten } i \in t} p_i \cdot \prod_{\forall \text{Kanten } i \notin t} (1 - p_i) \right)$$

Betrachten wir nun den speziellen Fall $p_i = \frac{1}{2}$ für alle $1 \leq i \leq n$, so ergibt sich

$$\begin{aligned} R(D_H) &= \sum_{\forall \text{FST } t \in D} \left(\frac{1}{2}\right)^n \\ R(D_H)2^n &= \sum_{\forall \text{FST } t \in D} 1 \\ &= \text{Anzahl der FSTs in } D \\ &= \text{Anzahl der Edge Covers in } G \end{aligned}$$

Da die Bestimmung der Anzahl der Edge Covers eines Graphen G NP-hart ist, ist somit auch die Bestimmung der DPR für ein DCS mit Sterntopologie NP-hart, sogar wenn $|FA_i| = 2$. □

Satz 4 *Die Berechnung der DPR für ein DCS mit Sterntopologie ist NP-hart. Dies gilt sogar, wenn es nur zwei Kopien von jeder Datei gibt.*

Der Beweis dieses Satzes wird über das „Number of Vertex Covers"-Problem (#VC) geführt. Sei ein ungerichteter Graph $G = (V, E)$ gegeben, dann ist #VC die Anzahl der Vertex Covers von G. Ein Vertex Cover ist eine Menge $K \subseteq V$, so dass jede Kante mindestens einen Endknoten in K hat. Die Bestimmung der Anzahl aller gültigen Vertex Covers ist NP-hart.[7]

Beweis Gegeben sei ein Graph $G = (V_1, E_1)$ mit $V_1 = \{v_1, v_2, \ldots, v_m\}$ und $n = |E_1|$. Hieraus konstruieren wir ein DCS $D = (V_2, E_2, F)$ mit einer Sterntopologie, wobei $V_2 = V_1 \cup \{s\}$, $E_2 = \{e_i = (s, v_i) | 1 \leq i \leq m\}$ und $F = \{f_i | \forall i \in E_1\}$. Sei weiter $FA_i = \{f_j | \text{für alle Kanten } j \text{ inzident zu } v_i \in V_1\}$ und $H = F$. Aus der Konstruktion von D ist nun leicht ersichtlich, dass es von jeder Datei genau zwei Kopien gibt, sowie dass jedem Vertex Cover von G exakt ein FST in D entspricht.

Die DPR von D kann daher wie folgt ausgedrückt werden:

$$R(D_H) = \sum_{\forall \text{FST } t \in D} \left(\prod_{\forall \text{Kanten } i \in t} p_i \cdot \prod_{\forall \text{Kanten } i \notin t} 1 - p_i \right)$$

Betrachten wir nun wieder den speziellen Fall $p_i = \frac{1}{2}$ für alle $1 \leq i \leq n$, so ergibt sich

$$\begin{aligned} R(D_H) &= \sum_{\forall \text{FST } t \in D} \left(\frac{1}{2}\right)^n \\ R(D_H)2^n &= \sum_{\forall \text{FST } t \in D} 1 \\ &= \text{Anzahl der FSTs in } D \\ &= \text{Anzahl der Vertex Covers in } G \end{aligned}$$

Da die Bestimmung der Anzahl der Vertex Covers eines Graphen G NP-hart ist, ist somit auch die Bestimmung der DPR für ein DCS mit Sterntopologie NP-hart, sogar wenn es nur zwei Kopien von jeder Datei gibt. □

3.4 DCS mit Baum-Topologie

Satz 5 *Die Berechnung der DPR für ein DCS mit Baumtopologie ist NP-hart.*

Beweis Wie die Sätze 3 und 4 zeigen, ist es im Allgemeinen NP-hart die DPR für ein DCS mit Sterntopologie zu berechnen. Da ein DCS mit Sterntopologie lediglich ein Spezialfall eines DCS mit Baumtopologie ist, nämlich eines Baumes mit Baumtiefe 1, ergibt sich hieraus zwingend, dass die Berechnung der DPR für ein DCS mit einer Baumtopologie NP-hart ist. □

3.5 DCS mit 2-Baum-Topologie bzw. serien-paralleler Topologie

Ein 2-Baum wird rekursiv wie folgt definiert:

1. Der vollständige Graph K_2 ist ein 2-Baum.
2. Gegeben sei ein beliebiger 2-Baum G mit $n \geq 2$ Knoten und (u, v) sei eine Kante von G. Durch das Hinzufügen eines neuen Knoten w, sowie den zwei Kanten (u, w) und (v, w) entsteht ein 2-Baum mit $n + 1$ Knoten.

Satz 6 *Die Berechnung der DPR für ein DCS mit 2-Baum-Topologie ist NP-hart.*

Beweis Der Beweis basiert auf der Reduktion einer beliebigen Instanz einer Stern-Topologie auf eine 2-Baum-Topologie. Gegeben sei der DCS-Graph $D = (V, E, F)$ mit $V = \{s, v_1, \ldots, v_n\}$ und $E = \{(s, v_i)|1 \leq i \leq n\}$. Durch Konstruktion des DCS-Graphen $D' = (V, E', H)$ aus D mit $E' = E \cup \{(v_i, v_{i+1})|1 \leq i \leq n - 1\}$ ergibt sich, dass D' ein 2-Baum mit $n + 1$ Knoten ist. Unter der Annahme, dass alle hinzugefügten Kanten $(v_i, v_{i+1}), 1 \leq i \leq n - 1$ in D' die Zuverlässigkeit $p = 0$ besitzen, folgt $R(D_H) = R(D'_H)$ für beliebiges $H \subseteq F$. □

Satz 7 *Die Berechnung der DPR für ein serien-paralleles DCS ist NP-hart.*

Beweis Ein 2-Baum ist ein maximaler serien-paralleler Graph, d. h. ein serien-paralleler Graph ohne Schlingen und parallele Kanten.[11] Da die Berechnung der DPR für ein DCS mit 2-Baum-Topologie NP-hart ist, folgt automatisch, dass die Berechnung der DPR für ein DCS mit serien-paralleler Topologie ebenfalls NP-hart ist. (Siehe auch Satz 6. D' ist ein serien-paralleler Graph.) □

4 Fazit

Die Zuverlässigkeit eines verteilten Programms in einem DCS drückt sich in der Wahrscheinlichkeit aus, mit der ein Programm, welches auf einer Vielzahl von Verarbeitungselementen läuft und Kommunikationsverbindungen zu anderen Verarbeitungselementen benötigt, erfolgreich ausgeführt wird. Dabei determiniert die zu Grunde liegende Netzwerktopologie zu großen Teilen die Zuverlässigkeit.

Die Berechnung der DPR eines allgemeinen DCS ist (nach Satz 1) NP-hart. Sogar wenn das DCS eingeschränkt wird auf Baum-, Stern-, 2-Baum- oder serien-parallele Topologien bleibt die Lösung dieses Zuverlässigkeitsproblems NP-hart.

Literatur

[1] M. O. Ball, J. S. Provan, und D. R. Shier. Reliability covering problems. *Networks 21*, Seiten 345–357, 1991.

[2] Michael R. Garey und David S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and company, 1979.

[3] John Hopcroft, Rajeev Motwani, und Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, second edition, 2001.

[4] V. K. Prasanna Kumar, Salim Hariri, und C. S. Raghavendra. Distributed program reliability analysis. *IEEE Transactions on Software Engineering, Vol. SE-12, 01/1986*, Seiten 42–50, 1986.

[5] Min-Sheng Lin und Deng-Jyi Chen. The computational complexity of the reliability problem on distributed systems. *Information Processing Letters 64 (1997)*, Seiten 143–147, 1997.

[6] o. V. *Duden Informatik: ein Sachlexikon für Studium und Praxis.* Mannheim: Bibliographisches Institut & F.A. Brockhaus AG, 1988.

[7] J. S. Provan und M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing, Volume 12*, Seiten 777–778, 1983. ⟨URL: `http://locus.siam.org/SICOMP/volume-12/art_0212053.html`⟩ – Zugriff am 17.12.2004.

[8] A. Rosenthal. A computer scientist looks at reliability computations. *Reliability and Fault tree Analysis SIAM*, Seiten 133–152, 1975.

[9] Andrew Tanenbaum und Marten van Steen. *Verteilte Systeme: Grundlagen und Paradigmen.* München: Pearson Education Deutschland GmbH, 2003.

[10] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing, Volume 8*, Seiten 410–421, 1979. ⟨URL: `http://locus.siam.org/SICOMP/volume-08/art_0208032.html`⟩ – Zugriff am 17.12.2004.

[11] P. Winter. Steiner problems in networks: A survey. *Networks 17*, Seiten 129–167, 1987.

CD

Auf der CD sind alle im Quellenverzeichnis angegebenen Quellen aus dem Internet enthalten (jeweils im PDF-Format). Der Dateiname setzt sich hierbei aus der Kombination des Namens des ersten Autors und dem Erscheinungsjahr zusammen (*Name-Jahr*). Zusätzlich befindet sich auf der CD diese Arbeit im PDF-Format.