Dependable Systems
(VU 182.712)

Praktisches Ubungsbeispiel SS2015:
"Zuverlaessigkeitsmodellierung mit *sharpe*"

Datum der Laboruebung: 18.06.2015

Matr. Nr.	**Name**	**Email**
1228774	Schieber Constantin	Constantin.Schieber@outlook.com
1325582	Pacheiner Peter	Peter.Pacheiner@gmail.com
1226314	Hofer David	e1226314@student.tuwien.ac.at

1 Abstract

Ein Computernetzwerk soll durch Markovketten modelliert werden um es auf Fehlersicherheit zu überprüfen.
Konkret soll die Mean Time to Failure (MTTF) und die Verfügbarkeit zwischen 2 Systemen evaluiert und verglichen werden. Eines ohne Redundanz und eines mit. Zusätzlich soll ein Vergleich zwischen den Kosten für die beiden Systeme durchgeführt werden.

2 Executive Summary

Es wurden 4 verschiedene Systeme mit Markovketten modelliert:

- Das einfache Modell
- Das redundante Modell mit absorbierenden Zustand
- Das einfache Modell mit Reparaturkanten
- Das redundante Modell mit Reparaturkanten

In Punkt 4.3 wird die MTTF anhand der ersten 2 Modelle simuliert und das Ergebnis ausgewertet. In Punkt 5.3 wird anhand der letzten 2 Modelle die Availability des Systems simuliert und das Ergebnis ausgewertet.
Unter Punkt 6 werden die Kostenunterschiede zwischen redundanten und einfachen Systemen diskutiert und eine Grenze, ab der sich ein redundantes System für unser Problem auszahlt, berechnet.

3 Aufgabenstellung

Für beide Systeme gelten die folgenden Fehlerraten (λ) und Reparaturraten (μ).

- $\lambda_R = 10^{-4}/Std.$ für Rechner
- $\lambda_N = 2 * 10^{-5}/Std.$ für Switches
- $\mu = 10^{-2}/Std.$ für Beide

  Um funktionsfähig zu sein benötigt das Netzwerk mindestens einen Switch und drei Rechner.

4 Mean Time to Failure

4.1 Einfaches System

Das Computernetzwerk besteht aus drei über einen Switch verbunden Rechnern.
Um die MTTF zu evaluieren reicht es das System durch 2 Zustände zu modellieren.
Einer stellt den funktionierenden Zustand (0) dar, der andere den Fehlerzustand (1). Die Wahrscheinlichkeit für einen Übergang in den Fehlerzustand setzt sich dann aus $3*\lambda_R+\lambda_N$ zusammen. Diese Vereinfachung kann vorgenommen werden da jeder Ausfall, egal welcher Art, in einen Fehlerzustand führt. Die Reparaturrate muss nicht berücksichtigt werden da wir uns im Moment ausschließlich für die MTTF interessieren.
Das System kann sowohl graphisch als auch in Textform beschrieben werden:

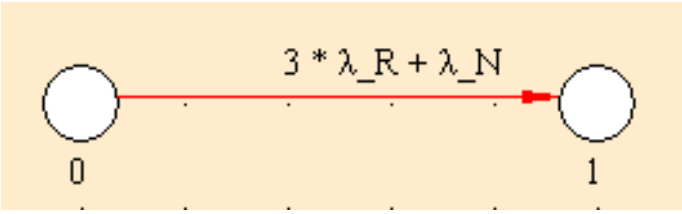

Abbildung 1: Einfaches System in *sharpe* modelliert

```
*Description of the simple model
markov simple
0 1 lambda_N + 3*lambda_R
end
```

4.2 Redundantes System

Das redundante Computernetzwerk besteht aus 4 Rechnern und 2 Switches was die Ausfallwahrscheinlichkeit gegenüber dem einfachen System deutlich verringern sollte.
Statt 2 Knoten werden nun 5 Knoten benötigt um alle Übergänge korrekt abzubilden.

Das System ist in den Knoten 0,1,2,3 funktionsfähig, Knoten 4 ist der Fehlerzustand (siehe Grafik 2).

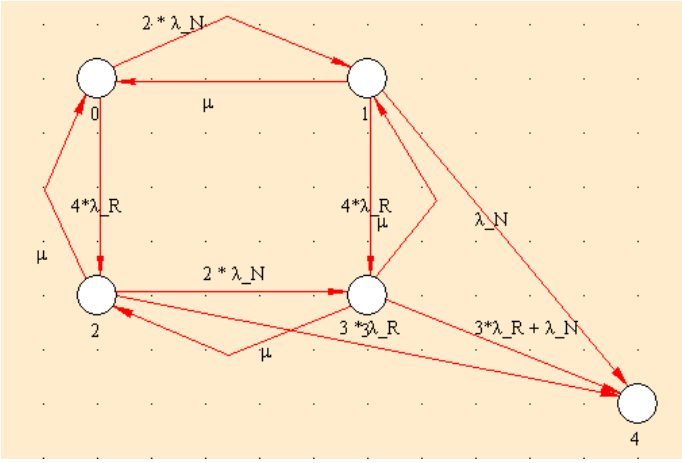

Abbildung 2: Redundantes System in *sharpe* modelliert

```
*Description of the redundant model
markov redundant

2 0 mu
3 2 mu
1 3 mu
1 0 mu
0 1 2*lambda_N
0 2 4*lambda_R
2 3 2*lambda_N
1 3 4*lambda_R
3 4 3*lambda_R + lambda_N
1 4 lambda_N
2 4 3*lambda_R
end
```

4.3 Vergleich der beiden Systeme

Um die beiden Systeme vergleichen zu können muss nun auch die Simulation derselbigen durchgeführt werden. Dies geschieht durch folgende Befehlsseqünz in *sharpe*:

```
bind
lambda_N      2/100000
lambda_R      1/10000
mu            1/100
end
*Description of the simple model
markov simple
0 1 lambda_N + 3*lambda_R
end
*Starting probability
0 1.0
end
*Description of the redundant model
markov redundant
2 0 mu
3 2 mu
1 3 mu
1 0 mu
0 1 2*lambda_N
0 2 4*lambda_R
2 3 2*lambda_N
1 3 4*lambda_R
3 4 3*lambda_R + lambda_N
1 4 lambda_N
2 4 3*lambda_R
end
*starting probability
0 1.0
end
* calculate MTTF
expr mean(simple)
expr mean(redundant)
```

Daraus resultieren die folgenden Ergebnisse:

```
/* Simple System with absorbing state */
mean(SIM):    3.1250e+003
```

```
/* Redundant System with absorbing state and with some repair edges */
mean(RED):    8.8540e+004
```

```
/* System with absorbing state and absolutely no repair edges */
mean(RED):    5.7678e+003
```

Das bedeutet dass das einfache Modell nach ca. 3125 Stunden ausfällt und das Redundante erst nach 8854 Stunden. Man sieht dass sich durch das hinzufügen der Redundanz der Zeitraum bis zum Ausfall deutlich erhöht. Zusätzlich zu den beiden genannten Modellen wurde aus Interesse ein drittes Modell getestet welches zwar redundant ist aber absolut keine Reparaturkanten besitzt. Wie man sehen kann ist bei diesem System eine Zeit von 5767 Stunden bis zum ersten Ausfall angegeben. Das ergibt 'nur' 2632 Stunden mehr als bei einem einfachen System. Man kann also deutlich sehen dass ein redundantes System zwar die Ausfallsicherheit erhöht, ohne ausreichend schnelle Reparatur aber schnell das selbe Schicksal wie ein einfaches (und damit billigeres) System erleidet.
Um die Unterschiede zu verdeutlichen wurde ein Diagramm (Abb. 3) erstellt. In diesem kann man gut erkennen dass der Unterschied zwischen dem Einfachen und dem redundanten System ohne Reparaturkanten vernachlässigbar klein ist.

Mit *sharpe* ist es möglich sich die Cumulative Distribution Functionßu berechnen. Mit dieser Funktion lässt sich bestimmen welchen Wert eine Zufallsvariable x maximal annimmt (zwischen 0 und 1).
Der Aufruf inklusive Ausgabe der Werte sieht dann so aus:

```
* evaluation (cumulative distribution function)
cdf(redundant)

* Start End Inc
eval(redundant) 0 500000 5000
end
```

Diese Funktion wurde auf die drei Systeme angewandt und sieht als Plot folgendermaßen aus:

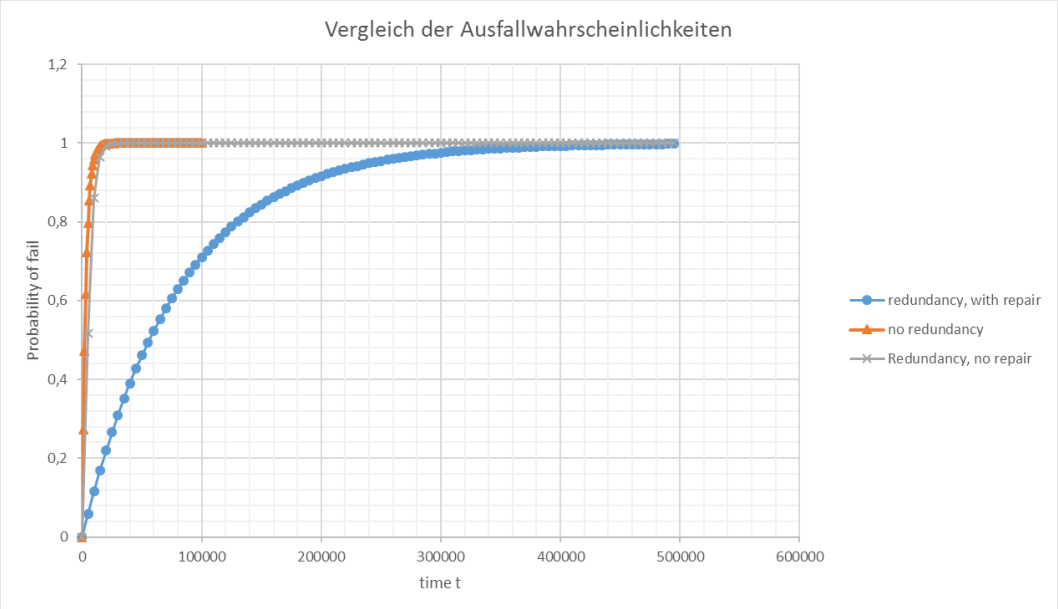

Abbildung 3: Unterschiedliche Strategien für Systeme

5 Availability

Im Gegensatz zur MTTF ist bei der Availability die Reparaturrate nun auch vom Fehlzustand aus zu berücksichtigen.

5.1 Einfaches System

Für das einfache System wird einfach eine neue Kante die zurück in den funktionierenden Zustand führt hinzugefügt.

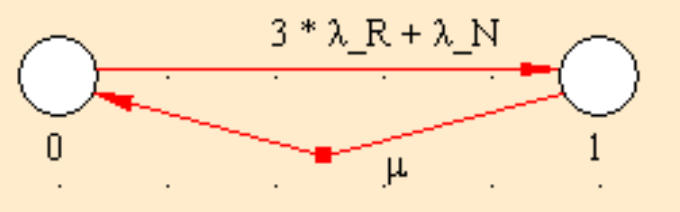

Abbildung 4: Einfaches System mit neür Kante in *sharpe* modelliert

```
*Description of the simple model with new repair edge
markov simple
0 1 lambda_N + 3*lambda_R
1 0 mu
end
```

5.2 Redundantes System

Im redundanten System reicht nun ein Fehlerzustand nicht mehr aus da wir aus diesem auch wieder zurück kehren können. Um sich den Zustand des restlichen Systems zu merken"bzw.. um deterministisch agieren zu können werden 3 neue Knoten eingeführt, die jeweils als Fehlzustände fungieren und über eine Kante mit dem letzten funktionierenden Zustand verbunden sind.

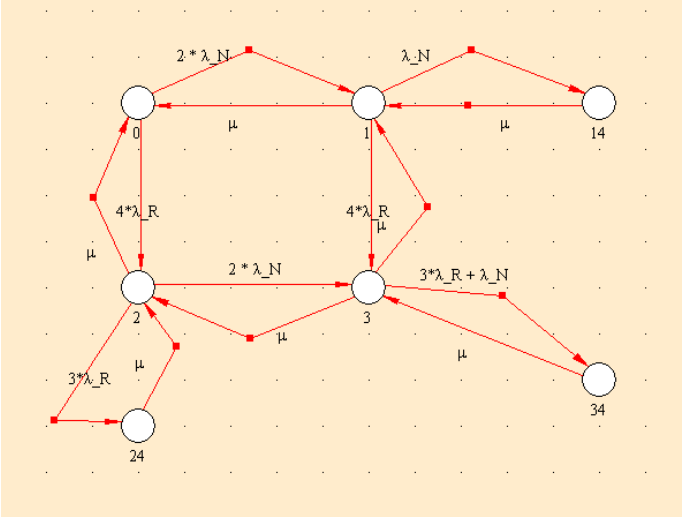

Abbildung 5: Redundantes System in *sharpe* modelliert

```
*Description of the redundant model with new failstates
*and edges for repairrates
markov redundant
2 0 mu
3 2 mu
1 3 mu
1 0 mu

0 1 2*lambda_N
0 2 4*lambda_R
2 3 2*lambda_N
1 3 4*lambda_R

*Create new failstates for a deterministic approach
3 34 3*lambda_R + lambda_N
1 14 lambda_N
2 24 3*lambda_R

*Add new edges for the repairrates
34 3 mu
14 1 mu
24 2 mu
end
```

5.3 Vergleich der beiden Systeme

Der Vergleich der beiden Systeme wird mit dem Befehl *prob(SYS, NODE)* ausgeführt. Dieser errechnet die Zustandswahrscheinlichkeit für eine bestimmte *NODE* im angegebenen System.
Im einfachen System reicht es die Wahrscheinlichkeit berechnen zu lassen mit der sich das System im gültigen Zustand befindet.
Im redundanten Fall ist es einfacher die Wahrscheinlichkeit der Fehlzustände berechnen zu lassen, diese zu addieren und von 100% abzuziehen.

Der Aufbau der Befehlsabfolge sieht so aus:

```
bind
lambda_N      2/100000
lambda_R      1/10000
mu            1/100
end
*Description of the simple model
markov simple
0 1 lambda_N + 3*lambda_R
1 0 mu
end
*Starting probability
0 1.0
end
*Description of the redundant model
markov redundant
2 0 mu
3 2 mu
1 3 mu
1 0 mu
0 1 2*lambda_N
0 2 4*lambda_R
2 3 2*lambda_N
1 3 4*lambda_R
3 34 3*lambda_R + lambda_N
1 14 lambda_N
2 24 3*lambda_R
34 3 mu
14 1 mu
24 2 mu
end
*starting probability
0 1.0
end
*calculate Availability
expr prob(simple, 0)
expr 1-(prob(redundant, 24) + prob(redundant, 34) + prob(redundant, 14))
end
```

Und das Ergebnis dann folgendermaßen:

```
prob(simple, 0):
    9.6899e-001
```

```
1-(prob(redundant, 24) + prob(redundant, 34) + prob(redundant, 14)):
    9.9872e-001
```

Wie man sieht beträgt die Wahrscheinlichkeit dafür dass sich das einfache System in einem gültigen Zustand befindet 96.899%.
Die Wahrscheinlichkeit in einem redundanten System beträgt 99.87%.

6 Kosten

Abschließend soll noch berechnet werden, ab welchem Verhältnis von Ausfallkosten zu Normalbetriebskosten des Systems, die fehlertolerant erweiterte Systemvariante zu bevorzugen ist. Laut Angabe ist der Normalbetrieb der erweiterten Variante 2.5 mal so teuer wie der Normalbetrieb der einfachen Variante. Während sich ein System im ausgefallenen Zustand befindet, verursacht es laufende Ausfallkosten, die sich auf ein Vielfaches der Betriebskosten der einfachen Variante belaufen.

In den weiteren Berechnung werden die folgenden Bezeichnungen verwendet:

$K_{Normalbetrieb}$: Kosten für den Normalbetrieb des einfachen Systems
$K_{Ausfall}$: laufende Ausfallkosten, im Fall dass sich das System im Zustand „Ausgefallen“ befindet.
K_{GS} : Gesamtkosten für die simple Systemvariante
K_{GR} : Gesamtkosten für die redundante Systemvariante
C : Vielfachheit der Kosten bei Systemausfall im Vergleich zum Normalbetrieb der einfachen Variante.
$C = \frac{K_{Ausfall}}{K_{Normalbetrieb}}$

$t_{Normalbetrieb}$: Zeit in der sich das System laut Spezifikation verhält
$t_{Ausfall}$: Zeit in der sich das System im Zustand „Ausgefallen“ befindet.

A_S : Verfügbarkeit des einfachen Systems
A_R : Verfügbarkeit des redundant aufgebauten Systems.

Die Gesamtkosten für ein System setzen sich aus den Betriebskosten während des Normalbetriebs und den Ausfallkosten während der Zeiten in denen sich das System im ausgefallenen Zustand befindet zusammen.

Damit ergeben sich die folgenden Gesamtkosten:

$$K_{GS} = K_{Normalbetrieb} * t_{Normalbetrieb} + K_{Ausfall} * t_{Ausfall}$$
$$K_{GR} = 2.5 * K_{Normalbetrieb} * t_{Normalbetrieb} + K_{Ausfall} * t_{Ausfall}$$

Die Ausfallkosten für das Systems bleiben gleich, unabhängig davon, welche Variante gewählt wird und betragen ein Vielfaches der Kosten des einfachen Systems im Normalbetrieb.

Wir können also weiter schreiben:

$$K_{GS} = K_{Normalbetrieb} * t_{Normalbetrieb} + C * K_{Normalbetrieb} * t_{Ausfall}$$

$$K_{GR} = 2.5 * K_{Normalbetrieb} * t_{Normalbetrieb} + C * K_{Normalbetrieb} * t_{Ausfall}$$

Die durchschnittlichen Zeiten für Normalbetrieb und Ausfall berechnen sich wie folgt:

$$Verfügbarkeit = \frac{Zeitdauer\,der\,Einsatzfähigkeit}{Zeitdauer\,der\,Einsatzfähigkeit + Zeitdauer\,der\,Nicht - Einsatzfähigkeit}$$

$$A = \frac{t_{Normalbetrieb}}{t_{Normalbetrieb} + t_{Ausfall}}$$

$$A(t_{Normalbetrieb} + t_{Ausfall}) = t_{Normalbetrieb}$$

$$t_{Ausfall} = t_{Normalbetrieb} * (\frac{1}{A} - 1)$$

Damit ergibt sich:

$$K_{GS} = K_{Normalbetrieb} * t_{Normalbetrieb} + C * K_{Normalbetrieb} * t_{Normalbetrieb} * (\frac{1}{A_S} - 1)$$

$$K_{GS} = K_{Normalbetrieb} * t_{Normalbetrieb} * (1 + C * (\frac{1}{A_S} - 1))$$

und

$$K_{GR} = K_{Normalbetrieb} * t_{Normalbetrieb} * (2.5 + C * (\frac{1}{A_R} - 1))$$

Die fehlertolerant erweiterte Systemvariante ist zu bevorzugen, sobald gilt:

$$K_{GR} < K_{GS}$$

Die Konstante berechnet sich daher zu:

$$(2.5 + C * (\frac{1}{A_R} - 1)) < (1 + C * (\frac{1}{A_S} - 1))$$

$$1.5 + C * (\frac{1}{A_R} - 1) < C * (\frac{1}{A_S} - 1)$$

$$1.5 < C * ((\frac{1}{A_S} - 1) - (\frac{1}{A_R} - 1))$$

$$C > \frac{1.5}{(\frac{1}{A_S} - 1) - (\frac{1}{A_R} - 1)}$$

$$C > \frac{1.5}{(\frac{1}{A_S} - \frac{1}{A_R})}$$

Setzen wir nur die vorher mit Hilfe von SHARPE berechneten Werte für die Availability ein erhalten wir:

$$C > \frac{1.5}{(\frac{1}{0.96899} - \frac{1}{0.99872})}$$

$$C > 48.82693$$

$$\frac{K_{Ausfall}}{K_{Normalbetrieb}} > 48.82693$$

Sobald also die Ausfallkosten des Systems mehr als das 48.83-fache der Normalbetriebskosten betragen, ist die fehlertolerant erweiterte Systemvariante zu bevorzugen.