P2P and Grid Computing
- Exercise 10 -
- C110 -

Michael Scholz (Matr. 1576630)
Ulf Gebhardt (Matr. 1574373)

H 10.1:

i)

Zustände: $S = \{s_0, s_1\} = \{u_{online}, u_{offline}\}$

Initialzustand: $I = (1, 0)^T$

Transitionsmatrix: $T = \begin{pmatrix} 0.85 & 0.05 \\ 0.15 & 0.95 \end{pmatrix}$

ii)

$P_2 = T^2 I$

Das Quadrat der Transitionsmatrix lässt sich einfach über das Falk-Schema berechnen. Im nächsten Schritt wird das Ergebnis der Matrixmultiplikation mit dem Initialisierungsvektor multipliziert (ebenfalls mittels Falk-Schema).

		0.85	0.05
		0.15	0.95
0.85	0.05	*0.73*	*0.09*
0.15	0.95	*0.27*	*0.91*

		1
		0
0.73	0.09	*0.73*
0.27	0.91	*0.27*

Somit ist die Wahrscheinlichkeit, dass U nach 2 Stunden immernoch online ist 0.73 73%.

iii)

Wir multiplizieren T solange mit sich selbst, bis das Element $t_{00} < 0.5$ wird.

k=1		k=2		k=3		k=4		k=5	
		0.85	0.05	0.85	0.05	0.85	0.05	0.85	0.05
		0.15	0.95	0.15	0.95	0.15	0.95	0.15	0.95
0.85	0.05	**0.73**	0.09	**0.634**	0.122	**0.5572**	0.1476	**0.495760**	...
0.15	0.95	0.27	0.91	0.366	0.878	0.4428	0.8524	...	...

Nach 5 Stunden fällt die Wahrscheinlichkeit, dass u immernoch online ist unter 0.5 50%.

iv)

Nach 19 Stunden fällt die Wahrscheinlichkeit, dass u immernoch online ist unter 0.05 5%. Diese Zahl wurde mittels folgendem Java-Code bestimmt.

```
public static void calculateMatrix(double[][] t, double percentage){
    int hour = 1;
    double[][] tmp = new double[2][2];
    double[][] sol = new double[2][2];

    tmp = t;

    while(true){
        hour++;
        for(int i = 0; i < t.length; i++){
            for(int j = 0; j < t[i].length; j++){
                sol[i][j] += tmp[i][j] * t[j][i];
            }
        }

        if(sol[0][0] < percentage){
            System.out.println("Hour: "+hour);
            System.out.println("Value: "+sol[0][0]);
            return;
        }
        tmp = sol;
        sol = new double[2][2];
    }
}


public static void main(String[] args) {
    double[][] bla = new double[2][2];
    bla[0][0] = 0.85;
    bla[0][1] = 0.05;
    bla[1][0] = 0.15;
    bla[1][1] = 0.95;

    calculateMatrix(bla, 0.05);

}
```

v)
Die Wahrscheinlichkeit, dass u immernoch online ist konvergiert gegen 3%.

<u>H 10.2:</u>
a)
i:
Ja, CAN hat eine reguläre degree distribution, da jeder Knoten alle seine Nachbarknoten kennt.
ii:
Ja, Gnutella baut solange Verbindungen auf, bis die Grenze der vom Nutzer definierten maximalen Anzahl an Verbindungen erreicht ist. D.h je besser der Rechner/Anbindungen/etc., desto mehr Verbindungen kann ein Nutzer aufbauen.
iii:
Nein, Chord besitzt immer eine reguläre degree distribution. Das ist durch die einzelnen Finger-Tables gewährleistet.
b)
Das linke Bild zeigt die in-degree distribution. Es ist zu erkennen, dass es sehr wenige ältere Knoten gibt, die eine große Anzahl an eingehenden Kanten besitzen. Das rechte Bild beschreibt die out-degree distribution. Es ist hier klar zu erkenne, das bei ca. 500 der Bucket des Clients vollgelaufen ist.
c)

$$E(D)=\sum(k*c*(1/k^a))=c*\sum(k*(1/k^a))=c*\sum(1/k^{a-1})$$

Wir schränken a ein mit 2 < a < 4 und können nun den Erwartungswert wie folgt einschränken:

$$c*\sum(1/k^3)<c*\sum(1/k^{a-1})<c*\sum(1/k)$$

$$\sum(1/k^3)<\sum(1/k^{a-1})<\sum(1/k)$$

$$1{,}2<c*\sum(1/k^{a-1})<\infty$$