Technische Universität München
Lehrstuhl Informatik VIII
Prof. Dr.-Ing. Georg Carle
Dr. Heiko Niedermayer
Cornelius Diekmann, M.Sc.

Netzwerkanalyse Sommersemester 2014
Assignment 1

Task 1 Deterministischer Netzwerkknoten (Theorie)

Wir betrachten einen Knoten in einem deterministischen Netzwerk. Der Netztechniker sagt Ihnen, dass jede Sekunde genau im Sekundenabstand 300 kleine Pakete ankommen. Genau eine halbe Sekunde versetzt kommt jeweils pro Sekunde ein großes Paket an. Es gibt zwei ausgehende Links A (in Richtung Alice) und B (in Richtung Bob). 150 kleine Pakete und das große Paket nehmen Link A und 150 kleine Pakete Link B. Die Verarbeitung eines kleinen Paketes dauert 5 ms, die des großen 50 ms. Die Pakete werden von den Links in FIFO-Ordnung (First In First Out) abgearbeitet.

a) Bestimmen Sie die Wartezeit (waiting time) und Durchlaufzeit (cycle time) der großen Pakete.

Lösung: 150*5 ms = 750 ms durch kleine Pakete, die großen Pakete kommen 500 ms später an. Daher 250 ms Wartezeit. Daher 250 ms + 50 ms Durchlaufzeit. Da unter 1s gibt es keine weiteren Effekte durch vorige Sekunden.

b) Bestimmen Sie die mittlere Durchlaufzeit (cycle time) der kleinen Pakete.

Lösung: 2 Pakete 0 ms, 2 Pakete 5 ms, ... 2 Packete $149 * 5ms$ = 75,5 * 5 ms
Wartezeit(kleine Pakete) = 372,5 ms
Durchlaufzeit(kleine Pakete)=Wartezeit + 5 ms = 377,5 ms

c) Die Auslastung (utilization) einer Bedieneinheit ist definiert als der Zeitanteil, an dem diese belegt ist. Wie groß ist die Auslastung von Link A?

Lösung: In 1 s, 750 ms Belegung durch kleine und 50 ms Belegung durch große Pakete. Daher 80 % Auslastung.

Task 2 Networkx (Programmierung)

In diesem Task üben wir etwas die Programmierung mit Networkx in Python. Auf der Webseite finden Sie die Datei graphpos2.py, welche hilfreiche Beispiele für die Aufgaben beinhaltet. Weitere Hilfen finden sich unter http://networkx.github.io/documentation/networkx-1.8/.

a) Erzeugen Sie mit networkx in Python den folgenden Graphen und visualisieren Sie ihn.

b) Geben Sie jeder Kante ein zufälliges Kantengewicht zwischen 1 und 5. Lassen Sie sich den Kürzeste-Wege-Baum von Knoten A ausgeben.

c) Bestimmen Sie mit Hilfe von networkx Center, Radius und Durchmesser (Diameter) des Graphen.

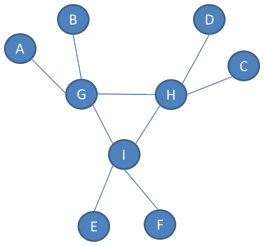

Figure 1: Graph

Lösung: aufg_1_2.py

Task 3 Gabriel-Graphen (Programmierung)

Auf den unteren Schichten im Netzwerk, z.B. im Falle von optischem Netzwerken im Backbone, finden sich v.a. Graphen, die die geographische Nähe von Knoten bei der Generierung von Links (edges) berücksichtigen. Beispiele für echte Netzwerke finden Sie z.B. unter http://sndlib.zib.de.

a) Erzeugen Sie mit networkx in Python einen Graphen mit 20 Knoten. Geben Sie jedem Knoten eine zufällige x- und y-Koordinate (das Gebiet habe die Größe 1000x1000).

b) Schreiben Sie eine Funktion, die als Input den Graphen bekommt und aus den Koordinaten der Knoten das für die Visualisierung benötige Positions-Dictionary ($Knoten \longmapsto [x, y]$) erzeugt. Das Positions-Dictionary ist der Output.

c) Erzeugen Sie nun die Kanten so, dass ein Gabriel-Graph entsteht. Bei einem Gabriel-Graphen haben zwei Knoten genau dann eine Kante, wenn im Kreis um den Kantenmittelpunkt kein näherer Knoten liegt. Visualisieren Sie den Graphen.

Mathematischer formuliert:
$\forall n, m \in G : (n, m) \in G \leftrightarrow \forall k \in G :$
$geographicDistance(k, center((n, m))) \geq geographicDistance(n, center((n, m)))$
wobei $center((n, m))$ den geographischen Mittelpunkt der Links (n,m) meint und $geographicDistance(n, m)$ den euklidischen Abstand zweier Knoten (Positionen) auf der Karte angibt.

Lösung: gabrielgraph.py