

# Architektur verteilter Anwendungen – Übung 1

## Aufgabe 1 (Basisimplementierung eines lokalen Knotens)

Implementieren Sie einen lokalen Knoten als einen Prozess, d.h. *kein Thread*. Der Knoten soll folgendes tun:

- 1) als Eingabe eine Datei lesen, die zu *IDs* Endpunkte als IP-Adresse und Portnummer zuordnet

Beispiel:

```
1 isl-s-01:5000
2 isl-s-01:5001
3 127.0.0.1:2712
...
```

- 2) als Eingabe (Kommandozeile) eine eindeutige *ID* erhalten
- 3) den zur eigenen *ID* zugehörigen Listen-Port *p* öffnen
- 4) sich aus den übrigen vorhandenen *IDs* drei Nachbarknoten wählen
- 5) Nachricht *m* auf Port *p* empfangen
- 6) Ausgabe der empfangenen Nachricht mit Zeitstempel
- 7) (nur einmal) die eigene *ID* an alle Nachbarn senden,
- 8) Ausgabe der gesendeten Nachricht mit Zeitstempel
- 9) gehe zu 5)

Die gesendeten Nachrichten sind Nachrichten der *Anwendung*.

Das zweite benötigte Nachrichtenformat sind *Kontrollnachrichten*.

Kontrollnachrichten sind vom Benutzer erzeugte äußere Ereignisse.

Nutzen Sie Kontrollnachrichten, um

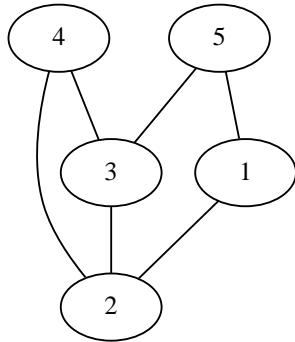
- einen Initiator festzulegen
- einen bzw. alle Knoten zu beenden

## Aufgabe 2 (graphviz-Format als Eingabe der Topologie)

*graphviz* ist ein Tool, mit dem man Graphen darstellen kann.

<http://www.graphviz.org/>

Eine Beispiel-Eingabedatei für den Graphen



sieht so aus:

```
graph G {
1 -- 2;
3 -- 2;
4 -- 2;
4 -- 3;
5 -- 1;
5 -- 3;
}
```

Erweitern Sie Ihr Programm aus Aufgabe 1 so, dass Sie eine solche *graphviz*-Datei als Eingabe einlesen und jeder Knoten die Nachbarn einliest, die zu seiner *ID* gehören.

## Aufgabe 3 (Zufälligen zusammenhängenden Graphen erzeugen)

Schreiben Sie ein Programm **graphgen**, das eine Eingabedatei wie in der vorigen Aufgabe zufällig erzeugt. Vorgegeben sollen Knotenanzahl  $n$  und Kantenanzahl  $m > n$  sein. Damit der entstehende Graph zusammenhängend ist, gehen Sie so vor:

```
for i=1 to n do
  choose random node j in {1,2,3,...,i-1}
  insert edge {i,j}
add edges until number of edges = m
```

Erzeugen Sie mit *graphviz* einige Beispielgraphen und überzeugen Sie sich, dass Ihr Code korrekt funktioniert.

#### Aufgabe 4 (Ausbreitung eines Gerüchts)

Unser erstes Experiment untersucht, wie sich ein Gerücht in einem sozialen Netzwerk ausbreitet. Nutzen Sie die zufälligen Graphen aus der vorigen Aufgabe, um per Kontrollmessage einem Initiator-knoten ein Gerücht zu „erzählen“. Der Initiator-knoten erzählt es an alle seine Nachbarn weiter. Erhält ein Knoten Kenntnis von dem Gerücht erzählt er es wiederum seinen Nachbarn weiter, außer natürlich dem, von dem er es hörte. Falls der Knoten das Gerücht schon kennt, erzählt er es nicht mehr weiter.

Ein Knoten *glaubt* das Gerücht, wenn er es von mindestens  $c$  Knoten gehört hat.

Untersuchen Sie

- 1) wieviele am Ende das Gerücht glauben, variieren Sie hierbei  $n$ ,  $m$ ,  $c$
- 2) erstellen Sie aus  $n$ ,  $m$ ,  $c$  eine Tabelle der Messergebnisse aus 20 Testläufen