

## 2. Übungsblatt

## Übung Peer-to-Peer-Systeme und Sicherheit (SS2009)

22. Mai 2009

**Abgabetermin:** Fr. 12.6.

**Übungstermin:** Do. 25.6.

Dipl.-Inform. Heiko Niedermayer

Lehrstuhl für Netzarchitekturen und Netzdienste

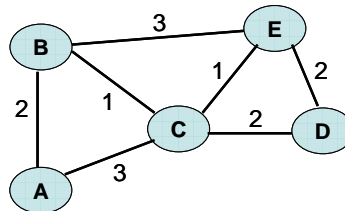
Technische Universität München

### Regeln:

Es sind insgesamt 5 Übungsblätter mit je 10 Punkten vorgesehen. Ein Blatt kann von 3 Studenten gemeinsam bearbeitet werden. Für die Übungscredits wird erwartet, dass Aufgaben für 30 Punkte sinnvoll bearbeitet wurden und dass 1x erfolgreich eine Aufgabe vorgerechnet worden ist.

### **Aufgabe 1 (2 Punkte) Clustering-Koeffizient C und charakteristische Pfadlänge L**

In dieser Aufgabe beschäftigen wir uns mit dem Clustering-Koeffizient und charakteristischen Pfadlängen.



Berechnen Sie den C und L sowie den Durchmesser des obigen Graphen.

Lösung:

A hat Nachbarn B,C  $\rightarrow C_A = 1 / 1 = 1$  (beide Nachbarn verbunden)

B hat Nachbarn A,C,E  $\rightarrow C_B = 2 / 3 = 0,67$  (2 von 3 möglichen Verbindungen)

C hat Nachbarn A,B,E,D  $\rightarrow C_C = 3 / 6 = 0,5$  (3 von 6 möglichen Verbindungen)

D hat Nachbarn C, E  $\rightarrow C_D = 1 / 1 = 1$  (beide Nachbarn verbunden)

E hat Nachbarn B,C,D  $\rightarrow C_E = 2 / 3 = 0,67$  (2 von 3 möglichen Verbindungen)

$$\rightarrow C = (1 + 2/3 + 0,5 + 1 + 2/3) / 5 = (2,5 + 4/3) / 5 = (15 / 6 + 8 / 6) / 5 = 23 / 30 = 0,766$$

$$A-B 2 \quad A-C 3 \quad A-D 5 \quad A-E 4 \quad B-C 1 \quad B-D 3 \quad B-E 2 \quad C-D 2 \quad C-E 1 \quad D-E 2$$

$$\rightarrow L = (2+3+5+4+1+3+2+2+1+2) / 10 = 25 / 10 = 2,5$$

$$D = 5$$

### Aufgabe 2 (2 Punkte) Clustering-Koeffizient C

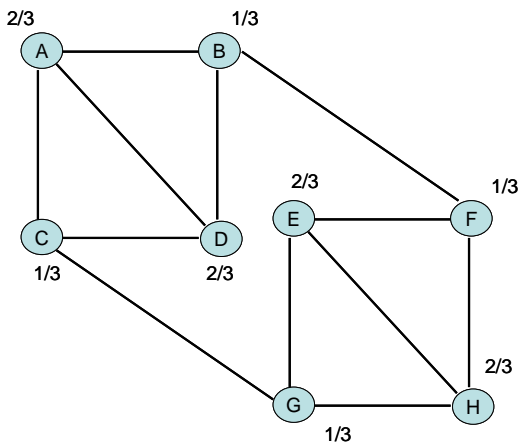
In dieser Aufgabe sollen Beispielgraphen mit einem bestimmten Clustering-Koeffizient gebaut werden.

- Geben Sie einen verbundenen Graphen mit 8 Knoten an, der etwa  $C=0,5$  hat. Zeigen Sie, dass dies auch wirklich der Fall ist, indem Sie C berechnen.
- Geben Sie einen Graphen mit 5 Knoten und mindestens 5 Kanten an, der  $C=0$  hat. Zeigen Sie, dass dies auch wirklich der Fall ist, indem Sie C berechnen.

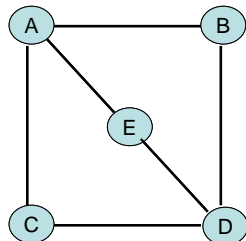
Lösung:

a)

z.B. mit offensichtlich  $C = 0,5$



b)



z.B.  $C$  aller Knoten ist 0, da Nachbarn immer unverbinden.  $\rightarrow C=0$

### Aufgabe 3 (2 Punkte) Key-based-Routing-API

Geben Sie in der Aufgabe jeweils kurz die Lösungsidee und entsprechend benötigte Funktionen in einem abstrahierten Pseudocode an, der unnötige Details und Längen vermeidet.

Nehmen Sie dabei an, Ihnen stünde ein strukturiertes Peer-to-Peer-Netzwerk zur Verfügung, das die Key-based-Routing-API implementiert hat:

- Schätzen Sie die Größe  $n$  des Peer-to-Peer-Netzwerks, in dem Sie mittels einer Nachricht in einem Key-Bereich die Knoten zählen und dann hochrechnen. Wie effizient ist Ihr Verfahren?
- Um Verluste proaktiv zu bekämpfen, verdoppeln Knoten mit Wahrscheinlichkeit  $p=2/\log(n)$  Nachrichten und senden die Zweite über einen anderen nächsten Hop. Wie viele zusätzliche Pakete erwarten Sie bei der Vorgehensweise?

Lösung:

a)

Idee: Sende Zähl-Nachricht über gewissen Bereich im Netzwerk je entlang der Nachfolger, dann hochrechnen mit

$$\text{Size\_estimate} = \frac{\text{Zählwert} * \text{Größe\_ID\_Raum}}{\text{Größe\_Intervall.}}$$

Starten mit `route(, COUNTMSG|myID|destID|count=0, successor)`. Eigenen Successor gegebenenfalls über Anfrage bestimmen, sofern der nicht bekannt ist.

```

void forward(key ⇔ K, msg ⇔ M, nodehandle ⇔ nextHopNode)
{
...
IF M is count message THEN
    Counter in M um 1 erhöhen
    nextHopNode = successor
    return
FI
...
}
void deliver(key ⇔ K, msg ⇔ M, nodehandle ⇔ nextHopNode)
{
...
IF M is count message THEN
    route(,COUNTREPLY|M, M.myID)
FI
IF M is count reply message THEN
    estimate_size = M.count * sizeFull_ID_range / d(M.destID,M.myID)
FI
...
}

```

b)  
In Forward-Upcall eine Verdopplungsroutine einbauen.

```

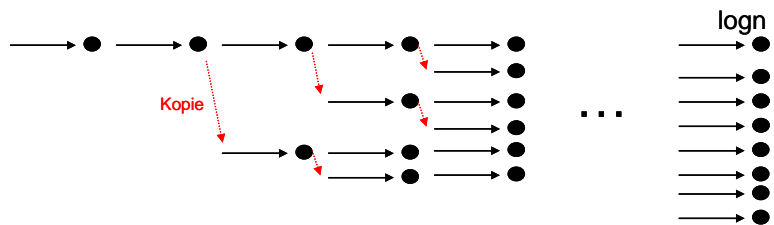
void forward(key ⇔ K, msg ⇔ M, nodehandle ⇔ nextHopNode)
{
...
IF random_U(0,1) <= 2 / log_2(size_estimate) THEN
    route(K,M,random_node from local_lookup(K,max_node,true))
FI
...
}

```

Abschätzung:

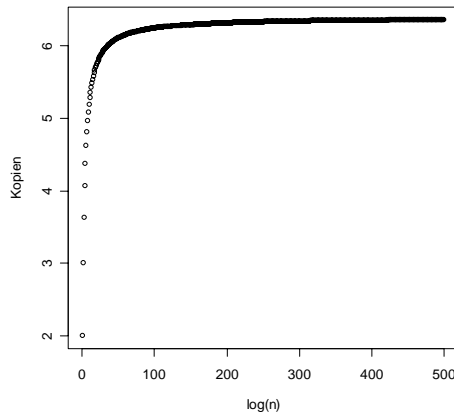
Pro Hop wird im Mittel aus einer Nachricht  $1 + p = 1 + \frac{2}{\log(\text{size\_estimate})} \approx 1 + \frac{2}{\log n}$

In jedem weiteren Schritt passiert genau dies wieder pro Nachricht. Gilt für jede kodierte Nachricht, dass sie sich trotz Alternativpfad echt einen Hop an das Ziel angenähert hat (der Weg für das kodierte Pakete also stets kürzer geworden ist), so starten wir mit einer Nachricht und für jeden der  $\log(n)$  Hops vergrößert sich die mittlere Nachrichtenzahl um den Faktor  $1+p$ . Schematisch:

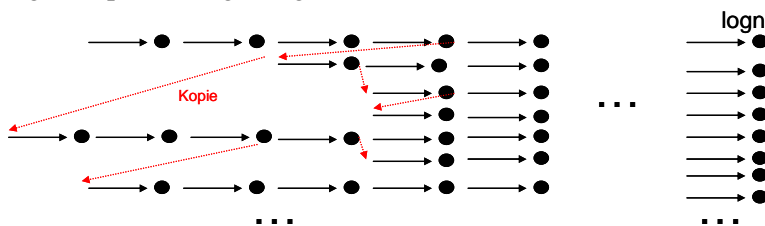


Gesamtzahl der am Ende parallel laufenden Pakete (im Mittel) =  $\left(1 + \frac{2}{\log n}\right)^{\log n}$

Zusätzliche Kopien (im Mittel) =  $\left(1 + \frac{2}{\log n}\right)^{\log n} - 1$



Gilt die Bedingung nicht, dass wir uns auch bei den Kopien in jedem Schritt echt ans Ziel annähern, so legen kopierte Wege längere Pfade zurück und werden so im Mittel häufiger verdoppelt.



Im diesem Fall besteht die Gefahr der Divergenz, da immer wieder Nachrichten mit längeren Weg zum Ziel entstehen können und diese wieder ggf. Nachrichten mit langem Weg und eher vielen Verdopplungen erzeugen. Die Grundidee, dass pro Nachricht etwa 2 Kopien erzeugt werden, wird hier massiv verfehlt und sehr wahrscheinlich das Netzwerk mit unbegrenzt vielen Nachrichten verstopft. Hier könnte ein Verdopplungsflag o.ä. Abhilfe schaffen, so dass nur die Originalnachricht verdoppelt werden kann.

#### Aufgabe 4 (2 Punkte) Verständnisfragen

Ein paar Fragen, die mit dem Wissen aus der Vorlesung beantwortbar sein sollten.

- Kennt der Tracker bei BitTorrent die Inhalte der über den Tracker verteilten Dateien?
- Angenommen, Peers wollten die Verteilung einer Datei sabotieren, indem sie im Schwarm fehlerhafte Subpieces versenden. Hat der Benutzer am Ende eine fehlerhafte Datei?
- Warum werden Power-Law-Netze als recht robust gegen zufällig Fehler angesehen?

Lösung:

a)

Nein, er kennt nur die Torrent-ID und beteiligte Peers.

b)

Nein, da die Hashwerte in der .torrent-Datei stehen und falsche Chunks somit verworfen werden. Die Performance leidet aber unter dem Angriff.

c)

Zufällig Fehler treffen mit hoher Wahrscheinlichkeit unwichtige Knoten und mit geringer wichtiger Knoten. Darüberhinaus wird noch angenommen, dass die unwichtigen Knoten weniger online und fehleranfälliger sind (z.B. Heimrechner am DSL statt Unirechner mit fester IP).

### **Aufgabe 5 (2 Punkte) Sensoren-Schwärme**

Bei trackerlosem BitTorrent wird eine DHT verwendet, in der ein Replica Set von 8 Knoten um die Torrent-ID herum den jeweiligen Torrent verwaltet. Jeder dieser Knoten hält Informationen über die Gesamtheit der Schwarm-Mitglieder. Bei einer Anfrage reicht es, wenn einer der Knoten mit einer Teilmenge antwortet. Die Anfrage nach anderen Schwarmknoten ist dabei ein häufiges Ereignis. Das alle Knoten im Replica Set betreffende Eintragen und Austragen ist entsprechend im Vergleich deutlich seltener.

Wir wollen nun ein Beispiel betrachten, wo diese Situation anders herum ist. Ein Schwarm von Sensoren verwalte seine Mitgliedschaft analog zu BitTorrent. Die Anfrage danach ist aber nun weniger dynamisch. Vielmehr schreiben die Knoten immer wieder aktuelle Sensor-Informationen oder Meldungen in die Schwarm-ID.

- a) Angenommen, diese Daten seien wichtig. Daher sollten sie nicht nur auf einem Knoten gespeichert werden, aber um Aufwand zu sparen auch nicht auf allen Replica-Knoten für den Schwarm. Geben Sie eine ressourcenschonende Alternative an, welche mit vereinzelt Knoten-Ausfällen umgehen kann. Anzugeben sind allgemeine Vorgehensweise sowie Schreib- und Lesevorgang.
- b) Nun soll der aktuelle Mittelwert bestimmt werden. Wie kann das in ihrem Verfahren geschehen.

Lösung:

a)

Es kann ausreichen ein kleineres Replica-Set zu wählen und Daten immer auf diesen zu speichern. Eleganter ist die Betrachtung des Replica-Sets als DHT, wo diese Daten gezielt abgelegt werden und dabei ein weiterer Index entsteht, der Anfragen danach gezielt ermöglicht. Da das Replica-Set nicht allzu groß ist, könnte die DHT vollvermascht sein und somit als „One-Hop-DHT“ (ODHT) arbeiten.

Schreiben:

Finde einen Knoten im Replica-Set. Dieser routet die Nachricht anhand der Art der Daten (z.B. welcher Sensor, Art der Daten) an das kleinere interne Replica-Set in der DHT der Replica-Knoten.

Lesen:

Finde einen Knoten im Replica-Set. Je nach Abfrage reicht es, einen Knoten des „internen“ Replica-Sets einer Art von Daten (z.B. eines Sensors, eines Datentyps) anzusprechen. Bei komplexeren Abfragen muss dann aber wieder gesammelt werden.

b)

Bei fester Replica-Größe könnten alle Knoten zur Torrentverwaltung ihre Werte aussummieren und an die anderen Fluten. In kleinen Netzen mit vielleicht 20 Knoten sollte das kein Problem sein.

Alternativ könnten die Knoten sich entsprechend zur Aggregation der Werte organisieren. Beispielsweise könnte eine Nachricht alle Knoten passieren und jeder Knoten schreibt seine Werte (Summe und Anzahl) in die Nachricht. Geht man von unterschiedlich großen Replica-Sets aus. So, könnte man auch den aktuellen ID-Bereich hochzählen und nur Werte aus dem nächsten Intervall jeweils dazuzählen.

Alternativ könnte jedes Replica-Set einen Verantwortlichen wählen und die Verantwortlichen organisieren sich in einem Baum, entlang dessen Kanten zur Wurzel hin die Aggregation gemacht wird (Summe der Werte, Anzahl der Werte).

Mittelwert ist stets = Summ der Werte / Anzahl der Werte