Exercise 1

Monday 16.5 2011
Hand-in: Monday 23.5. 2011 in lecture
Exercise: Thursday 26. 5. 2011

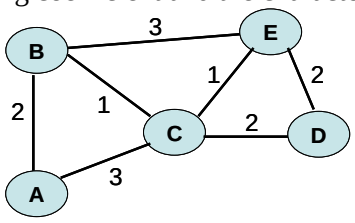
Exercises Peer-to-Peer-Systems and Security (SS2011)

Dr. Heiko Niedermayer
Lehrstuhl für Netzarchitekturen und Netzdienste
Technische Universität München

Rules: There will be five exercise sheets. You have to hand-in 70 % of the assignments, attend atleast 3 exercise courses and present a solution in the exercise course to get the 0.3 bonus.

Assignment 1 Clustering-Coefficient C and characteristic path length L

This assignment is about the clustering coefficient and the characteristic path length. Here is the graph.



Determine C, L as well as the diameter of the graph.

Solution:

A with neighbors B,C → C_A = 1 / 1 = 1 (both neighbors connected)
B with neighbors A,C,E → C_B = 2 / 3 = 0,67 (2 of 3 possible connections)
C with neighbors A,B,E,D → C_C = 3 / 6 = 0,5 (3 of 6 possible connections)
D with neighbors C, E → C_D = 1 / 1 = 1 (both neighbors connected)
E with neighbors B,C,D → C_E = 2 / 3 = 0,67 (2 of 3 possible connections)

→ 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 A-C 3 A-D 5 A-E 4 B-C 1 B-D 3 B-E 2 C-D 2 C-E 1 D-E 2

→ L = (2+3+5+4+1+3+2+2+1+2) / 10 = 25 / 10 = 2,5

D = 5

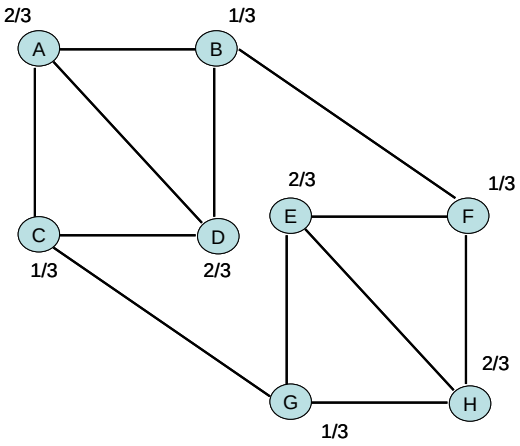
Assignment 2 (2 Points) Clustering-Koeffizient C

In this assignment you should create an example graph with certain properties.

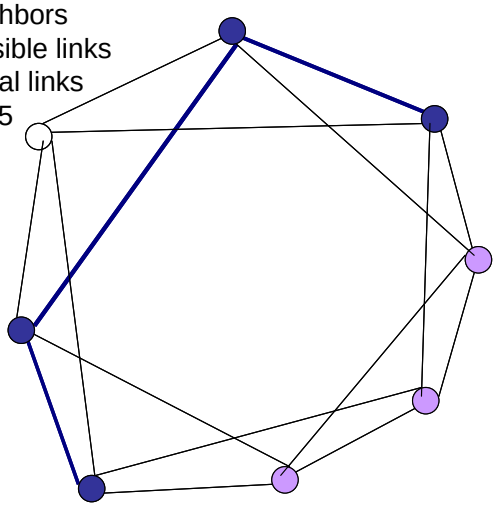
a) A connected graph with 8 nodes and C=0.5 (approx.). Prove your claim by calculating C.
b) A connected graph with 5 nodes and at lest 5 links and C=0. Calculate C for the graph.

Solution:

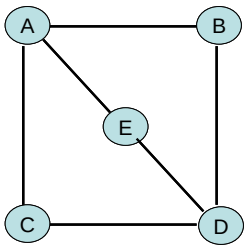
a) E.g. with apparently C = 0,5



4 neighbors
6 possible links
3 actual links
C = 0,5



b)



E.g.

or even a circle, because 5 connections are already sufficient.

C_v of all Nodes is 0, because none of its neighbors are connected. → C=0

Assignment 3 P2P Protocol

A protocol for an unstructured network. Each node joins via some node it knows. Then it operates as follows:

- Every 10 s the node asks a neighbor for 10 other nodes. For each of the up to 10 nodes in the reply the node computes a random number. With probability p=25 % it will contact the node and create a connection, and add it to the list of known nodes. Otherwise, it ignores the node.
- Every 5 s it will contact a neighbor to see if it still exists. If not, the connection to the node will be closed and the neighbor will be removed from the list of known nodes.

Questions:

a) What is the probability that none of the 10 nodes is contacted?
b) Give an example for a problem of this protocol. How could you fix it?

Solution:

a) The node has to decide 10 times not to connect its possible neighbors, each time with $p_{no\_connection}$ = 0,75.

0.75^(10) = 0.0593 = 5.63 %

b)

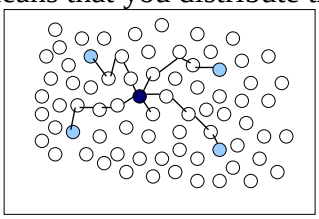
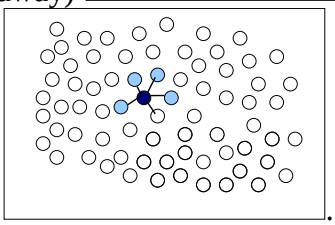
- The node has no additional contacts yet and his only known node leaves the net. → no connectivity.
- There is no stopping constraint for requesting new nodes or establishing connections to them. As a consequence, too many connections will be established, so it becomes difficult to maintain the net.
- If a node only gets to know nodes that will leave soon and the average amount of nodes remains small→ there is always the risk to loose connectivity with the net.
- Slow assembly of neighbourhood. One could say, the cache is being filled more slowly than necessary.
- ….

Assignment 4 CoolSpotsMunich I

Assume that the CoolSpotsMunich network is an unstructured network just as in the first slides of chapter 1.2 where the peers form an unstructured network and each peers stores its own data locally. Give an example on how to optimize the lookup for spots near your location in the network with unstructured methods (Do not add search structures!).

Solution:

Some potential options:

- Cache answer in network after or during reply.
    - at local node
    - replicate in the network (either on 2-3 neighbors on nodes on the search path where it was found)
    - requires softstate (delete replica after some time) or delete old or unpopular entries when the cache of a node overloads
    - State: depends on popularity and request rate, limited by cache size
    - This can be combined with the other approaches for replication and storage given in the following.
- A spot is replicated randomly on nodes that are at least 2 hops away in the graph, cache answers to your requests. This means that you distribute the replicas (blue) more over the graph (here 3 nodes away)  than with positioning them near the source (dark blue)  .
    - This reduces the distance of the spot to all nodes in the graph as the spots as spots are stored (more or less) all over the graph. The coverage improves.
    - State: O(1) more storage
- A spot is replicated to neighbors with spots close to the GPS coordinate of the spot. This requires that neighbors know what kind of spots their neighbors know of.
    - Also: Prefer links to candidate nodes with spots at similar GPS coordinates.
    - During lookup: prefer neighbors with spots close desired GPS coordinate.
    - While nodes always keep their own spots, they might move the replicated spots further down to neighbors of their own that have stored closer other spots. It seems to be desireable that the new holder of the replica gets into contact with the source to allow for easier maintenance.
    - This also requires that sources re-publish their spots to replicas every once in a while.
    - The basic goal of this idea is to provide a DHT-like situation of clusters of spots with similar coordinates, yet it does this in an imperfect way. This approach is related to the classic Freenet routing.

Assignment 5 CoolSpotsMunich II

Just like in Assignment 4, use the description from the beginning of chapter 1.2 if you need information about the network's operation.

a) If you look for a bakery, are you looking for nodes or for items?
b) If you want to find spots that your friends recommend, are you looking for nodes or for items?
c) What information do you need to add into the system so that you can detect that someone is your friend?
d) Based on your proposal for c), how can you optimize the discovery of friends in the network?

Solution:

a) Item
b) Nodes
c) Identities of people (nodes), a datastructure to state friendship (potentially stored in the network instead of only your local machine)
   Alternatives: node and people Ids are different, add lookup structure to find node ID / locator (= IP, port) for people ID

d) Lookup ID of the nodes belonging to the friends (alternative options: hold data in network)
   Alternatives: if node and people Ids are different, then first look up people Id and then the corresponding node ID or use the locator