Prof. Dr. Thorsten Strufe
Stefanie Roos
{strufe,roos}@cs.tu-darmstadt.de

Fachgebiet Peer-to-Peer Netzwerke
Fachbereich Informatik
Technische Universität Darmstadt

Exercise 9

for **Peer-to-Peer Networks** - winter term 2012/2013

(18.12.2012)

Deadline for submission: 15.01.2012 1:30 PM

Guidelines

- Exercises annoted by **G#** are intended to be discussed and solved in class without grading, whereas exercises annoted with **H#** are supposed to be solved in groups and be handed in for grading.
  This does not mean, that ungraded exercises are less important.
- Please submit your solutions until the beginning (1:30 PM) of the next exercise in the coming week. Solutions can either be dropped in the letterbox in front of A110 or handed in personally on the beginning of the exercise. **Electronic submission is no more allowed!**
- Note that points are only given if your solution is clearly legible. Unreadable submissions will not be rated! (Machine written submissions are allowed)
- Written assignments are to be solved in groups of 2-3 participants while programming assignments have to be done in groups of four to six participants.
- Always annotate your solutions on the handed in sheet with names and matriculation numbers. If you have privacy concerns, you are allowed to omit your name and tell it to us personally.
- Please subscribe to the mailing list:
  `https://mail.rbg.informatik.tu-darmstadt.de/mailman/listinfo.cgi/p2p-lecture-ws12`
- By submitting any processed exercises or program code you hereby commit to the "Grundregeln der wissenschaftlichen Ethik am Fachbereich Informatik" (see also `http://www.informatik.tu-darmstadt.de/de/sonstiges/plagiarismus/`).
  This especially means, that you should always write in your own words. However if you use external materials, you have to cite them correctly.
  We will not accept solutions that only rely on literal citations.

G#9.1 Network Resilience

Consider the graphs *G* and *H* in Figure 1 and 2, respectively.

a) Random Failures

How likely is it that the failure of one random node disconnects G or H?

b) Intentional Failure

Attackers generally try to remove nodes with the highest degree from a network assuming that they are the most influential. Does removing one node with the highest degree work in case of *G*, respectively *H*? If not, does it work to remove some other node?

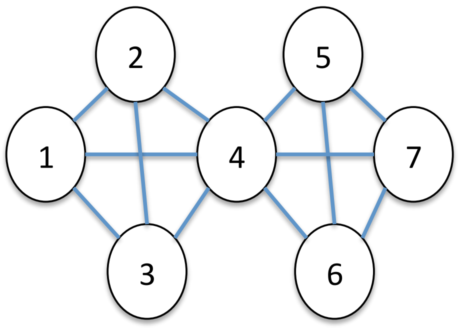

Figure 1: G

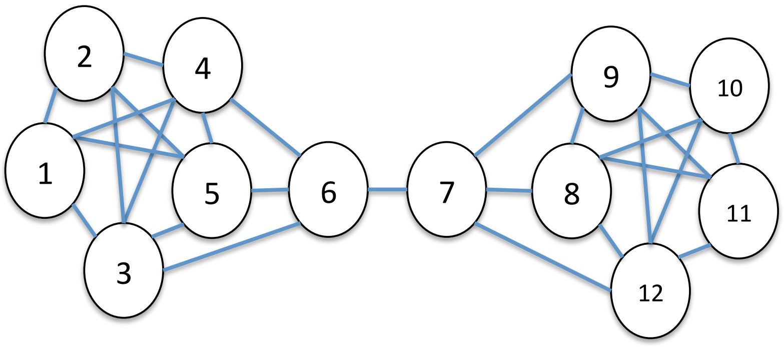

Figure 2: H

c) Laplacian Matrix

For a bidirectional graph with n nodes, the adjacency matrix A has entries $a_{ij} \in \{0,1\}$ where $a_{ij} = 1$ if there is an edge between node i and node j, and $a_{ij} = 0$ otherwise.
The degree matrix D is a diagonal matrix where d_{ii} is the degree of the node i.
Consider the Laplacian matrix $L = D - A$. Show that L always has eigenvalue 0. What does the multiplicity of the eigenvalue 0 (i.e. the dimension of the corresponding eigenspace) tell you about the graph?
Look at the eigenvectors for the second smallest eigenvalue for

$$v_G = (-0.408, -0.408, -0.408, 0, 0.408, 0.408, 0.408)^T$$

and

$$v_H = (0.631, 0.631, 0.587, 0.587, 0.587, 0.367, -0.367, -0.587, -0.587, -0.631, -0.631, -0.587)^T$$

of G and H, respectively. Interpret the i-th entry with regard to the importance of node i for connectivity.

H#9.1 CAN (11 P.)

a) Partitioning of CAN's identifier space (9 P.)

Assume a 2-dimensional CAN identifier space on $[0,32)^2 = [0,32)$ x $[0,32)$. Consider the case when nodes 1 to 10 consecutively join the overlay after randomly choosing an identifier. The identifiers selected by each node in two different scenarios are given in Table 1 & 2. When joining the overlay, a node selects a random identifier. After it has found its zone, it receives either the lower of the right part of that zone. Both the new node and the old owner of the zone change their identifiers to be the center of their new zones.

Draw the correct partitioning of the identifier space for both scenarios after the 10 nodes have joined the overlay. Label each partition with the corresponding node id and the number of the node. Additionally write down the neighbors for node 6 and 10.

number	1	2	3	4	5	6	7	8	9	10
id	(17/3)	(20/10)	(15/31)	(7/2)	(27/5)	(2/3)	(16/18)	(21/19)	(25/11)	(7/2)

Table 1: Identifiers chosen by the nodes during join in scenario 1

node	1	2	3	4	5	6	7	8	9	10
id	(30/7)	(1/9)	(10/26)	(17/3)	(30/20)	(14/5)	(29/22)	(13/11)	(3/14)	(1/1)

Table 2: Identifiers chosen by the nodes during join in scenario 2

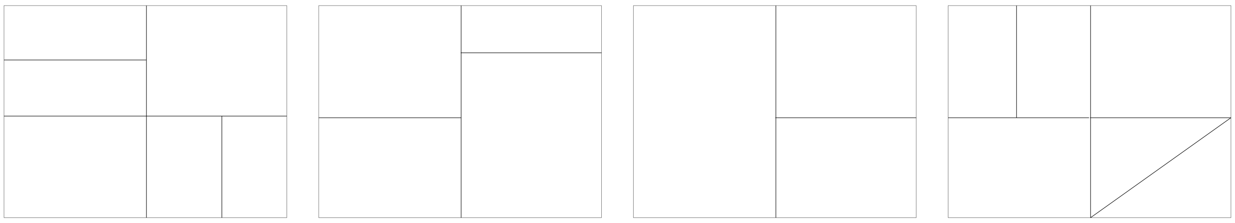

Figure 3: Partitions of CAN networks? (a), (b), (c), and (d) partitions from left to right.

b) Incorrect Partitioning of Identifier Space (2 P.)

Assume that CAN nodes successively join but do not leave the system. Which of the partitions shown in Figure 3 could be partitions of a CAN identifier space? Explain why.

H#9.2 Pastry/Tapestry (11 P.)

a) Routing Tables (4 P.)

Table 3 presents a list of Tapestry node-IDs. Create the neighbor table for node 1ef1 and 6f65 from those nodes. Use base-16 numbers. When several solutions are possible, choose the one with the largest ID.

06e4	0807	12e4	1494	1ef1	1fc0	21c4	2926	2958	2edb	31df	32f4
3711	378d	37d8	39f4	3cfa	4306	43b5	440f	4576	4844	4a67	4bcb
50ae	5601	57ee	5c79	5f7d	6a18	6f0b	6f65	7457	74c0	7504	7f21
849a	8a62	8ba8	8d96	9022	9302	95bf	9b86	9dc0	ac05	af58	b0d5
b253	b364	b621	b6c5	bb0f	c061	c673	cbfe	ce63	d0d9	f517	fb51

Table 3: Tapestry Nodes

b) Leaf Nodes (3 P.)

In Pastry, a node additionally stores L leaf nodes. This are the $L/2$ closest nodes in each direction on the ring (in case of $L = 2$ this corresponds to the successor and predecessor in Chord). Using again Table 3, what are the leaf nodes for 1ef1 and 6f65 when $L = 4$? How many nodes to fail at least, so that the underlying ring topology is destroyed (for arbitrary L)?

c) Routing (2 P.)

During routing, a node first checks its leaf set if it contains the target node, otherwise the message is forwarded to a node in the neighbor table with the longest common prefix with the target.
How is a message from 1ef1 to 6f0b routed? What kind of links are used?

d) Back-up nodes (2 P.)

In Tapestry, nodes usually select two back-up nodes for each entry (if present). Imagine that the same principle is used for Chord. Which nodes should be chosen as back-up, so that the entries are correct after a node failure?
Why do you think is this concept not implemented in KAD?