Unstructured P2P (11 points)

This homework is focused to establish an unstructured P2P overlay network and expand it with gossiping and naming service. Please read the theory behind naming and communication in P2P.

The solution for this homework must be uploaded to OLAT until Monday, **23.04.2018**, at **08:00**, and will be evaluated on 24th of April!

Implement an application which is capable of interacting with other peers in a P2P-like fashion. To save time, implement only features actually requested or required to support those. Read the full assignment and think before implementing features.

Each node within the network maintains a ServerSocket instance bound to some (arbitrary) port number. Further on, each node maintains a set of ***n*** (e.g. *n* = 3) known nodes (node *table*). Each entry within this set should consist of an IP address and a port number that can be used to contact remote nodes. When a peer is started, the user specifies the IP/port number of another node to initialize the node table, unless it is the first peer. In this case the table is left empty. Periodically (e.g. every 3 seconds) each node is picking a random node entry within its table. Let *Node A* be the node picking an entry, which is addressing *Node B*. *Node A* establishes a connection to *Node B* and requests *Node B*'s table. *Node B* responds with its own set of known nodes (table). Upon receiving *B*'s entries, *Node A* is merging its local and the received table and removes duplicates, as well as its own address from the result (to avoid having a self-reference within the node table). Finally, a random subset of ***n*** entries is picked to be the new table for *Node A*. In case *Node A* is not able to establish a connection to *Node B* (since it is offline or left), it (*Node A*) removes the entry for the *Node B* from its table and picks another entry until there is no further entry left.

Process the following steps:

1. Implement P2P Network *management* (add, remove, update table)
   Demonstrate the proper operation of your implementation by creating a network of $\sim 3 * n$ nodes, followed by removing existing and connecting additional nodes. Find a way to present the propagation of the information of entering and leaving/dying nodes throughout the network. How can you solve a possible problem of segregating the network? How can you join these separated parts? **(4 points)**

2. Describe an "effective" way of *gossiping* (propagating a one-to-(almost)all message on top of your network). For example, update a value of a parameter only at *Node A* and propagate that update. Assume that another update can happen at *Node A* even before all other nodes see that update. Try to keep the number of messages required to be forwarded low while still providing a "best effort" guarantee that messages are delivered to all nodes.
   Demonstrate the propagation of your message within a moderately sized network involving $\sim 3 * n$ nodes (*n = 100* or *n = 50*, or less if your machine does not support). Evaluate the gossiping in three tables with two columns *i*) *Step* and *ii*) Reached nodes for each step, by using three values of fraction *s* (*s = 0.1*, *s = 0.5* and *s = 0.9*) of neighbor nodes that will remain ignorant of an update. Why gossiping does not guarantee that all nodes will get the update? **(4 points)**

3. Assign a *name* to every peer within the network. Describe a name-resolution protocol on top of your network infrastructure. For instance, somebody wants to contact peer 'XY' – it therefore needs its IP / port number. How can it be obtained? Consider the possibility of 'XY' not being present or reachable within the network.
   Implement your lookup protocol by adding a corresponding method and demonstrate its proper operation. **(3 points)**