Home Assignment (Model checking)

Title: **P2P overlay**

Advisor: **Dániel Élő**

Problem Description

A file managing network consists of nodes. In order to belong to the network and to be able to exchange files, a node has to register itself with some of the nodes that already belong to the network.

Every node has a list called *active view* that contains references to nodes which have registered themselves on this node. The network is built-up if the size of the active view of every node is at least 1 (even if the graph representing the connections of nodes is not connected). Every node can have at most *k* others in its list. Being in the active view is a symmetric relation, so if *q* is in the active view of *p*, it means that *p* is also in the active view of *q*. The nodes have two states: *active* and *busy*. The nodes are in active state by default and they become busy if they start to process a message (*JOIN_REQUEST, FORWARD_JOIN*, see below). After processing the message is finished, they return to active state.

A node *p* signals its intent to connect to the network by sending a request *JOIN_REQUEST* to node *q*. If *q* is currently busy, it sends a message to *p* and ignores its request. If this happens, *p* must find another node. If the receiving node q was in active state, it changes to busy and adds the applicant to its active view. If *q*'s list would grow greater than k with this record, it deletes a random node from its list with sending the message *DISCONNECT* to it (because of the symmetric behavior of this relation it also gets deleted from the other node's list).

Then the receiving node *q* sends a *FORWARD_JOIN* message to every node in its active view that is in active state. This message contains the parameters of the applicant node and a *TTL* value. After sending the messages, the receiving node returns to active state.

If a node *n* is in active state when receiving a *FORWARD_JOIN* message, it can process it in two ways:

1. If the *TTL* is 0 or the size of its active view is 1, *n* registers the to-be-connected *p* in its own view (if it was not present earlier) and sends a message about this to *p*;
2. Else it decreases the *TTL* value with one, then selects a random active node from its own active view to send a *FORWARD_JOIN* message (like described above) with the updated *TTL* value.

Of course during processing *n* becomes *busy*, and after handling the message (either way) it becomes *active* again.

If the active view of a node becomes empty for some reason (e.g. multiple *DISCONNECT* messages) it has to connect to the network again. In the case of an active view of a node growing larger than k, it has to delete a node from its list with a *DISCONNECT* message.

Your task is to model the building process of a network and answer the questions regarding the requirements below.

Requirements to check

Do an analysis on a finite system consisting of 5 nodes. Parameter *k* is 3, starting *TTL* is 1. You can assume that the number of nodes in the system is constant.

Prove the satisfiability of the requirements below using temporal logic expressions and model checking (in case of unsatisfiability explain the reasons in detail with a counterexample)! Show and explain a short example/counterexample where possible!

1. The system has a reachable state where the network is built up.
2. There is no state where any node has 3 elements in its list.
3. Deadlock in the model is possible only if every node is connected to the network.
4. **Optional task (not needed, even for an excellent mark):** If requirement 3 is not satisfiable, give a proposal for fixing the protocol (integrating the proposal into the model is not necessary).