Design Plan - group 18

Group members **Kristian Krok**, **Laura North**, **Aksel Linros** and **Niko Häppölä**.

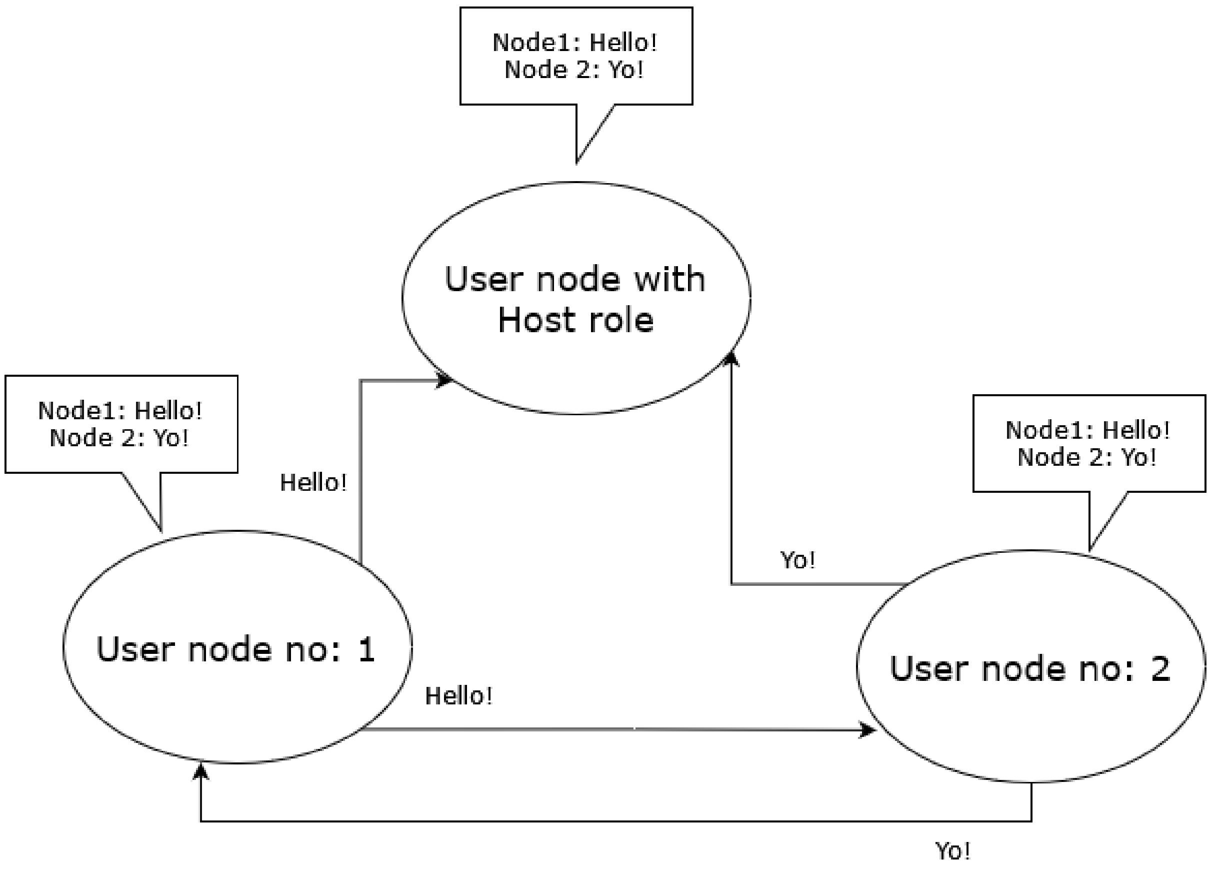
Norris Chat System is an instant messaging system for *Distributed Systems* -course in University of Helsinki.

Description of the topic

In this project we will create a distributed chat system, where users can broadcast messages to every other user in the network. All users are free to join and leave at any time, and will see the messages sent in the same order as everyone else. Receiving messages happens only when user is online, and the system does not save any previous messages.

The project will be implemented with Python, KMV + QEMU (for virtual machines) and possibly FastAPI or RabbitMQ as the message broker.

Architecture sketch



Nodes

A user is a single node in the system. There are no centralized servers, so the whole network consists of user nodes. All the nodes are pluripotent, so depending on the situation they can take either of these roles:

1. User node. The normal node type that can send and receive broadcasts from the network. When a new user node connects to the network, it will be given a unique identifier by the host, which is shown to all other nodes when a message is sent.

2. Host node. This is a special role for one user node that is needed for initializing connection of a new node. When a request to join the chat system comes, the host node will respond with information about all the nodes in the network, so that the new node can initialize communication paths. There must always be at least one host node, and the first node to start the network automatically gets assigned this role. In addition, the host node has all the user node capabilities, and from the usage point of view does not differ from a user node.

Messages

In the chat message, a node sends a message containing a timestamp, its unique identifier and the message.

The system aims to deliver all chat messages in chronological order. Should a message arrive for a node later than another message with an earlier timestamp, the system will arrange their order according to their respected timestamps.

The messages will be delivered using HTTP-protocol.

Other comments

The number of users isn't limited. The system aims to allow arbitrary number of simultaneous users. However, a significant number of simultaneous messages may prove to be difficult to the system to handle. This will be something we'll be looking further in the development.

The system will not provide a feature for the user to delete or edit messages that have already been sent. The system will not persist a message log, hence a fresh user will only see messages that have been sent after they have joined the system.