# Introduction

For this assignment we have proposed to build a distributed chat system. What we are trying to accomplish is a chat system that allows the nodes to broadcast messages to other nodes that have joined the chat group (peers). However, the chat server is distributed amongst all nodes that are currently running, with a super node used to alert other nodes of new nodes who have entered and introduce the new nodes that have entered to the environment and notify them of all existing members. This project is intended to make use of Scala, Scala FX, and AKKA.

# The Physical model

The proposed physical model of the distributed system is based on a peer-to-peer architecture. Our physical model consists of nodes. Every node is able to directly communicated with other nodes. Among of all the nodes, one of the nodes will be a super node. New node is required to join to a supernode in order to be in the same environment. Initially, a node is required to join its own to become a super node. Figure 1 shows the system architecture diagram.

# The Interactive model

Our distributed system is primarily consisting of four components, `ActorSystem`, `Supernode`, `Node` and `Display`. `ActorSystem` is responsible to handling the remote networking part of the system. It is subscribed to remote event such as `DisassociatedEvent` and `DeadLetter`, which then allow it to inform the nodes about the status of remote node. `Supernode` is responsible for passing the information of the chat system environment such as users and rooms. `Node`, on the other hand, plays the role of both client and server, handling all the core functionalities of the chat system, including keeping track of the users, rooms and messages. `Display` is responsible for managing the Graphical User Interface (GUI) of the system.

Our interaction model is based on asynchronous distributed system. Our system has no bounds on process execution speeds, message transmission delays and clock drift rates. This is primarily due to the fact that our system is relying on the Internet network to communicate with each other.

Figure 2 shows the communication protocol sequence diagram of the overall distributed chat system.

Note that in the end of the sequence diagram, when a supernode is offline, every node will received a `DisassociatedEvent` message. Each node will in turn check if they are the first node in the kept record, if they are the first node, they will send a `BecomeSupernode` message to itself and become the new supernode. The new supernode will then send `NewSupernode` message to every node in the environment to inform the changes.