* + Explanation how scalability and reliability can be guarantee in the program.

Reliability are critical in the design of most computer systems, especially distributed systems. Our distributed communication system is using the peer to peer architecture. For example, in a network of 5 Nodes, 1 of the Nodes will be assigned as the Supernode where the Supernode is responsible for relaying the state of Nodes in the network and handling new incoming Node. To ensure reliability of the system, if the Supernode disconnects or crashes, all the current Node will receive a notification on it and our program will assign the Supernode responsibility to another node. Doing that can enable a new Node to join the chat without the original Supernode. If there is no new Supernode, then a new Node is unable to the join the chat. The way we assigned a Supernode once the original Supernode crashes or disconnects is by choosing the name alphabetically in an ascending order. By assigning a new Supernode, the new Supernode will broadcast itself to every Node in the network informing on it is the new Supernode. Hence with this implementation, no matter which Supernode or Node crashes or disconnects, it can still handle a new incoming Node who wants to join the chat.

The Supernode is responsible for relaying all the current connected Nodes to a new incoming Node. Hence every time a new Node wants to the join the chat, it will communicate with the Supernode. Once joining the chat, the Node do not need to rely anymore on the Supernode to communicate with other Node as they themselves will store a copy of all the current connected Nodes. Hence the Node can start a Chat with another Node without the help of Supernode. Even if the Supernode crashes, the Node is still able to chat with other Node as they have their own local copy. This implementation can ensure that the Nodes are able to communicate with each other even without the Supernode.

Scalability in important in a distributed system, especially in a communication system where it may contain a lot of users communicating with each other. A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and number of users. The reason for the peer-to-peer architecture style is because it is highly scalable, the Node is only required to do their own part of responsibility. Each Node is only responsible for sending their own message and receiving message that are directed towards them. The Node is not responsible for the message sending of another Node that is not related to them. This allows the system to scale up to allows many Nodes as there is not one single Node that are responsible for everything.

One area for concern is the Supernode and Node is they need to store the local copy of all the current connected Node in the chat system. However, this is not a problem as they are only store the ActorRef of all the current connected Node which only consume a small storage size. If there are 100000 connected users, meaning there is 100000 records in the local copy of the all the Nodes, it will be no problem as they consume a little storage space and the computing power of current devices are able to iterate through the list with ease. This implementation allows the chat system to scale up to many users without breaking the core functionality.