International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Using Peer to Peer Approach of Distributed Systems to Design a Chat Room Application

Kennevor D.G. Kharsyntiew

Department of Computer Science & Engineering, Saveetha School of Engineering, Shillong, India

Abstract: ***A distributed system is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a set of related tasks. A distributed system can be physically instructed by two ways: First, fully connected network peer- to –peer approach in which each of the nodes is connected to each other. Second, partially connected network in which a direct links exist between some but not all pairs of computers. The purpose of this paper is to design a chat room application which is based on the architecture of distributed system. This paper used a peer-to-peer approach to design the chat room application.***

1. Introduction

A chat application can be designed using different approaches of distributed system architecture. We will explain in details a peer-to-peer approach. Even though different models are possible there remains some functionality in common as shown in Fig.1.1.

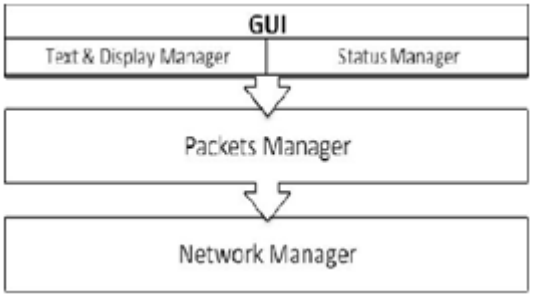

Figure 1.1: Architectural styles

In the above Fig. 1.1, Part A remains the same in all architectural styles which consist of GUI, text and display manager and status manager. Part B must be different in case of a peer-to-peer. In this paper section 3 represents the design scheme of architecture, section 4 develop a chat application use of a peer-to-peer architecture and how it work.

2. Objectives

This paper designs a chat room by using the architecture of distributed system which fulfils the following objectives:

(a) Design a Network scheme for peer-to-peer architecture.
(b) Use the peer-to- peer architecture of distributed system to design a chat room with following concepts:

- How does it work?
- Login and who is on-line?
- Ask for friend relation
- Chatting

3. Design Scheme

To design a chat room based on the peer- to– peer architecture of distributed system we will use the following managers as shown in Fig.2:

(a) Network manager. A network manager is responsible for listening to the network.
(b) Packets manager. The packet manager is responsible for classifying packets between system and normal chat messages.
(c) Text and display manager. The text and display manager is responsible for normal chat messages received from the packet manager.
(d) Status manager. This is responsible for status, login and friend messages.

Figure 2: Networks Managers

The most important thing is to run all these modules in separate threads because it is really important to deal with and display messages when the network manager is sending another one, etc. Fig. 3(b) showed the global shape of the architecture: peer-to-peer architecture.

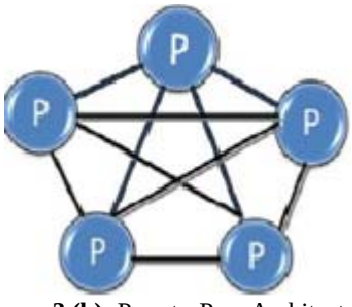

Figure 3 (b): Peer-to-Peer Architecture

We would try to explain more how the architecture can be developed and how we can deal with some aspects of the process such as login, discovery, friend relationships etc in

the next section of this paper.

(a) Peer-To- Peer Architecture

The first possibility to develop a chat application is the use of a peer-to-peer architecture of distributed system. A fully connected network peer to peer architecture is a network in which each of the nodes is connected to each other as shown in Fig. 1.3.

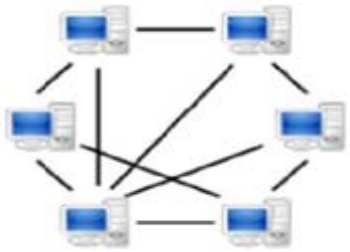

Figure 1.3: Node connection

The friends-list is modified when we add or remove a friend. This process will be explained in more details in our next section. The online-list is managed by the status manager. In a first time, we add peers into this list during the login phase. But, after that, we can use an eventually perfect failure detector (EPFD) to remove peers when they leave or they crash.

(b) Login and who is on-line?

We take an example with five peers (nodes) already connected on the network we are the sixth one arriving on the network as shown in Fig.1.5 and let's make some assumptions

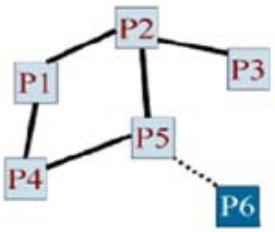

Figure 1.5: Example sixth one (node) arriving on the network

(a) How does it work?

We are working with a pure peer-to-peer network which is composed by peers and links between them only. We do not want to use the functionalities of super-peers to keep this system as basic and simple as possible. Every node must maintain two separated lists.

(i) One list containing names of friends (List<Friend1, Friend2, Friend3, ...,>).

(ii Another list containing all online peers that we know (List<[User1, IP, Port], [User2, IP, Port], [User3, IP, Port], ...>).

We assume that the discovery process is made easier thanks to a public place where connected peers can publish their point-of-entrance in the network. Whatever happens we need to know at least one entry point. We assume that there exists a (well managed) DHT (Distributed Hash Table) containing all credentials of registered people of this chat service.

Peer 6 chooses Peer 5 as entry point. Peer 6 will send a join message to Peer 5: **JOIN(Peer6, Password, IP, Port)** Figure 6 shows the behaviour of every node when a JOIN message arrives[6].

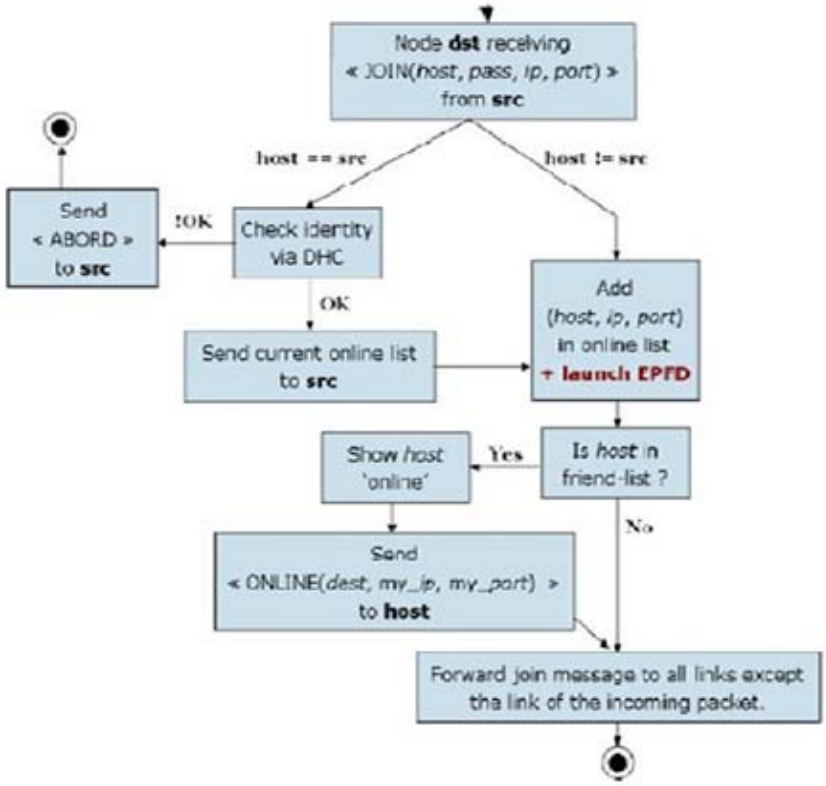

Figure 6: Behaviour model

In addition to that, we can imagine a way to avoid Join messages to loop forever in the network with a simple 'id' feature or something. Thanks to this login process, we can retrieve the list of already connected people and launch our EPFD on every host to detect when they leave or they crash. We are now ready to begin chat sessions with connected friends.

(c) Ask for friend relation

When the application that is running and the user is fully logged-in, we can simply search in the online list to friend new friends. Then, we can directly send an Ask-Friend message to the IP address and the port of the new friend. This kind of message is treated by the packets manager. If the new friend agrees with this new relationship, he replies with a Friend- Ok message to the original peer. If the new friend not wants to begin a relationship with the peer, he simply never replies to the message. This process is very simple thanks to the fact that we are in a peer-to-peer network, where every node can act as a server, client or both. The communication between peers can be done directly thanks to the information contained in the online-list.

(d) Chatting

In the same way than the friend messages, peers can simply exchange normal chat messages with each other thanks to the fact that they have all the information to join other peers. We can easily understand here the importance of the packets manager. Actually, when a packet arrives in a node, the packet manager can simply read the header of this packet to know if it is a system notification (friends, status, etc.) or a real chat message (containing text to display).

4. Conclusion

The peer-to-peer is not the best one for the login performance. We have to wait until our Join message reaches the farthest node to be fully connected to the network. If we have a global network with a high latency and a shortest path from us to the farthest node of about 50 hops, we must wait 50 times the mean latency of the network to be sure that everybody on the network is aware of our presence.

References

[1] Yang, B. H. & Garcia-Moline, "*Designing a Super-Peer Network*", Standford University, February 2002.
[2] Kwok, S., Lui, S., A License Management Model for Peer-to-Peer Music Sharing. International Journal of Information Technology & Decision Making, September, vol. 1, #3, pp. 541-558,2002.
[3] Lang, K. and Vragov, R. "*Using Experimental Methods to Evaluate the Effectiveness of Different Pricing Mechanisms for Content Distribution Over Peer-to-Peer Networks*." Americas Conference on Information Systems. 2005.
[4] Lechner, U. and Hummel, J. "*Business Models and System Architectures of Virtual Communities: From a Sociological Phenomenon to Peer-to-Peer Architectures*", International Journal of Electronic Commerce, 6, 3,2002. pp.41-53.
[5] E. Bonsma, C. Hoile. "*A distributed implementation of the SWAN peer-to-peer look-up system using mobile agents*". In Proc. of the AAMAS'02 Workshop on Agents and Peer-to-Peer Computing, Italy, 2002.
[6] Bibliography containing references on www.wikipedia.com
[7] Bibliography containing references on *Distributed Computing* can be found at: ftp:ftp.cs. umanitoba.ca/ pub/ bibliographies/ Distributed/ Oss er.html