A Peer to Peer Architecture Applied to Multiplayer Games

Felipe Rocha Wagner, Marcio Garcia Martins, Arthur Tórgo Gómez
Postgraduate Interdisciplinary Program in Applied Computing
University of Vale do Rio dos Sinos
São Leopoldo, Brazil
e-mail: feliperw@msn.com, marciog@unisinos.br, breno@unisinos.br

Abstract— **This article presents an architecture model developed on a Peer to Peer network, which gives support to develop multiplayer games that need to manage their peers connections and permissions. The model enables the development of multiplayer games, without the need of a dedicated server, as is observed in the most architectures. For this, the model offers a library that enables programmers access the network addresses, allowing them manage their peer connections and permissions. As results, of using this architecture model, we can be cite the reduction of the costs for developers of multiplayer games due to no need a dedicated server, and a greater flexibility to manage the peer connections and permissions by the use of the available library of the model.**

Keywords-manageable network; peer to peer; network address translator; transversal problem.

I. INTRODUCTION

The multiplayer games market comes growing in the last few years. It all started with the arcades and non-networked games as Spacewar![1] and Pong [1], what later evolved to become the networked online multiplayer games that we know today. Online multiplayer makes it easy to find people to play anytime and anywhere. Nonetheless, to connect many people in order to allow them to play together, we need a server or a Peer to Peer (P2P) mesh connection. Dedicated game servers are usually expensive for indie game developers. The cost arising from the use of dedicated game servers could be reduced using a P2P approach, when designing the game network. This is one point investigated in this work. P2P networks are not easy to build; there are many technological barriers that have to be broken to connect two or more peers in different private networks. Currently, this is a challenge for multiplayer games developers. Typically, each machine in a private network is hidden behind a public gateway, a public IP, with a Network Address Translator (NAT).

In this paper, we propose a P2P architecture applied to multiplayer games that need to manage their peers connections and permissions. The idea is to create a library which allows programmers access the network addresses, without a use of a server to manage the peer connections and permissions. The admin could define users groups in accordance with the dynamic of the multiplayer games. This way, we can have an admin being responsible for the network management. This admin, in a meeting application could mute users when someone is talking or divide the meeting at a moment when needed. In the same way, this admin could manage and balance the dynamic of game rooms.

This article is structured as follows. In Section 2, related works that were utilized to generate the architecture model proposed are presented. In Section 3, we introduce the architecture model and its modules and communication protocol. Section 4 presents the peers connection process. Finally, in the Section 5, the conclusion is presented.

II. RELATED WORK

In this section, we discuss on Super Peer in P2P Networks and NAT transversal problem that were utilized to generate the architecture model proposed in this paper.

A. *Super Per in P2P Networks*

According to Yang and Garcia-Molina [2], Super Peer is a node, in a P2P network, that works both as a server to a subset of clients as a peer in a network of Super Peers. Cao et al. [3] proposed a multi-level super peer based on P2P architecture designed to work in a hierarchical structure. The hierarchical model not only distributes the single points of failure in the network, reducing the chances of presenting a massive failure, but also helps in the development of servers or applications that are based on the same model. Based on this two the related proposals, we defined a variation of Super Peer. Our Super Peer (or Admin) is a peer in the network being responsible for the network management. It can work as a server for a set of clients connects to it, and, optionally, also works as a peer to the same set of clients. The Super Peer Network, that connects Super Peers with one another, will not be considered by this model.

B. *The Network Address Translator Transversal Problem*

The NAT is a table that translates private addresses to public addresses. The development of P2P applications utilizing the NAT has constraints, because it is not possible that two or more computing systems, in different private networks, send messages between them without a public address [4]. Some techniques that allow us to break this barrier appeared along the years [5]-[12]. The most common of these is the Hole Punching, which uses discovery and prediction techniques to find out the NAT mapping. More recently, some protocols as NAT-PMP [8], PCP [9] and UPnP [11][12] utilize communication protocol to configure the gateway and create a port-forwarding without the need of the user configuration.

III. Architecture Model

The architecture model was designed in a way to support message packets transferred and media streams between the network's peers. It also takes into account the existence of a Super Peer, which has the ability to manage the network configuration and permissions of other peers.

Every peer has its own network module, a set of configuration flags (that describe the permissions and communication rules), an ID number and a group; the latter two are defined by the network admin.

The model proposed can be described as a hybrid model of Client-Server and P2P. The admin user initially registers himself in a server and waits for connections from common users. After the connections are made, the users communicate directly with each other, without the need of a server that would increase the costs of this process. The only function of the server is to make possible the connection of the P2P network

A. *Modules*

The network modules are the core of all communication. Every peer has its own module, and every module is composed by two sub-modules, as shown in Fig. 1.

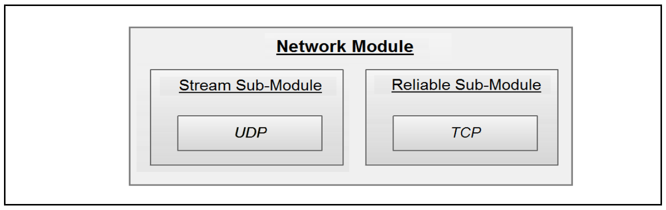

Figure 1. Network model.

The first one is a stream module for receiving and transmitting audio and video in real time using UDP, described by the RFC 768 [12]. The second one is responsible for delivering and receiving packets with network messages such as control and validation messages, and signals or important application messages. Those messages need to be sent, through a reliable connection without losing packets. Therefore, we chose to use TCP, described by the RFC 793 [13], which ensures the arrival of the packets in their destination [3][14].

To fully understand the sub-modules, we need to look at them separately. The Stream Sub-Module uses two UDP sockets, one to receive and other to transmit audio and/or video streams, as shown in Fig. 2.

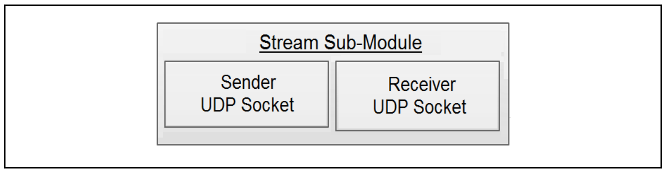

Figure 2. Stream sub-module.

On the other hand, the Reliable Sub-Module is composed of a listener responsible for receiving new connections and a list of sockets containing a functional socket for each connection sustained for peer, as seen in Fig. 3.

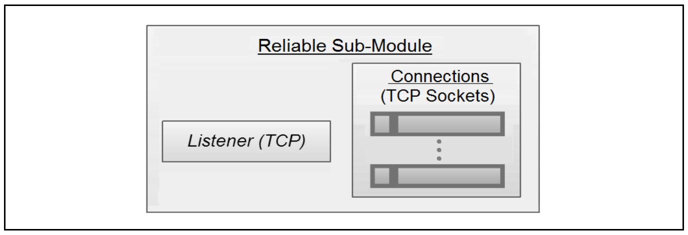

Figure 3. Reliable Sub-Module.

B. *Comunication Protocol*

According to Tanenbaum [15], a protocol is the set of rules and conventions governing communications between two or more computer systems. In the architecture model developed in this work, the admin can include himself in any kind of communication in his network, thus taking a full view of everything what is going on.

Users, in this architecture model, are divided into groups. Every user is connected to the admin, but not necessarily with the others users. The common users have connections with other common users, only when they belong to the same group. This way, the admin is able to send messages to any group in the network or even send a message from an user of a group to an user of another group. However, an user from a group is unable to directly send a message to an user of another group and vice versa.

In order to provide good performance to group system, the communication channels UDP and TCP have three ways to sending packets. The first one is the simple unicast, which is nothing more than the exchange of packets between two peers. The second one is multicast, and is used to send messages to a preset group of peer. At last, the third one is a broadcast, which is used to send messages to every peer with connection to the network.

The broadcast and multicast procedures can be simulated taking in account only the network mesh connections. Also, as observed in the Table I, broadcast and unicast procedures might present different behaviors according which the configurations and permissions of the peers.

TABLE I. BROADCAST & UNICAST BEHAVIOR

	Peers	Behavior
BroadcastM	Admin	Multicast → All
BroadcastG	User	Multicast* → Group
BroadcastR	User	BroadcastM Request (User → Admin)
Unicast	Admin-User	Unicast
Unicast	2 Users	Unicast or Multicast*

*Optionally might include the Admin as addressee

A user might broadcast messages in two ways. The first, we will call BroadcastG works as a multicast for the group the user belongs. The second occurs when a requests user to the admin to route a message for all network, including each single group, similar to a Broadcast Unknown Server (BUS) [15], that we are calling of BroadcastR.

To be able to use BroadcastR, an user must be enabled. A unicast between the common user and the admin will always be a simple unicast, but a unicast between two common users might behave as a multicast when the admin is included in the communication through the permissions and configurations of the peer. It is important to reinforce that the multicast and broadcast that we talk here might be simulated as a set of unicasts in its core.

C. *Network Packages*

The network messages can be wrapped in TCP or UDP packets, and are divided in two main groups: Common Messages and Control Messages. The difference between the two is a validation key of two bytes, appearing at the end of the packet header in the Control Messages, shown in the Table II.

TABLE II. HEADER OF WRAPPED PACKETS (EXT. = 0)

Offset (Bytes)	**1 Byte**			**1 Byte**	
	4 bits	**2 bits**	**2 bits**	**2 bits**	**6 bits**
0	Version	Type	Addressee Type	Ext. (= 0)	Reserved
2	Sender ID			Addressee ID	
4	Validation Key*				

*Present only in Control Messages

It is important to highlight that the validation key has the purpose to avoiding cheating in the network. The header starts with a four bits version number, matching the bits 0001. Next, we have two bits that define the type of message according to Table III.

TABLE III. MESSAGE TYPES

2 Bits Value	**Message Type**
00	Common Message
01	Common Message (Stream)
10	Control Message
11	Connection Message

The next two bits represent the addressee type, and define what will the Addressee ID corresponding, as follow. Addressee type: equal to zero (bits: 00) corresponds to a user; equal to one (bits: 01) corresponds to a group; equal to two (bits: 10) corresponds to a broadcast message; and equal to three (bits: 11) corresponds to a system message. After that, we have other two bits, which are used to establish the extension of the Sender and Addressee IDs as 2n Bytes, where n is the extension value.

The next 6 bits are reserved and should be ignored. The Bytes in sequence, should be construed according to the extension value. In case of the extension value is zero, the third Byte represents the Sender ID, and the fourth Byte represents the Addressee ID: which must be translated according to the Addressee Type value.

Only the admin has permission to send broadcast messages to the network. The users might request to the admin to send a broadcast message. If the users have the right permissions, the admin will work as a BUS sending the messages to all the users connected to him. To make a broadcast request, the common user must send an unicast message to the admin with its Addressee Type set as broadcast and the Addressee ID set to zero. It is up to the admin accepts or declines the request.

The stream transmission is equivalent to a common message, once there is no need for any validation of the frames arrival, what could cause delays in the transmission. We can stream audio, video or both (mux). To send and receive streams we must use an encoder and a decoder that will be responsible for processing the data. In this fashion, the codec or mux to be used is the responsibility of the application or of game developer.

IV. CONECTION PROCESS

To connect the peers in a network, we must follow a connection protocol. The connection protocol for this architecture model is defined in Fig. 4.

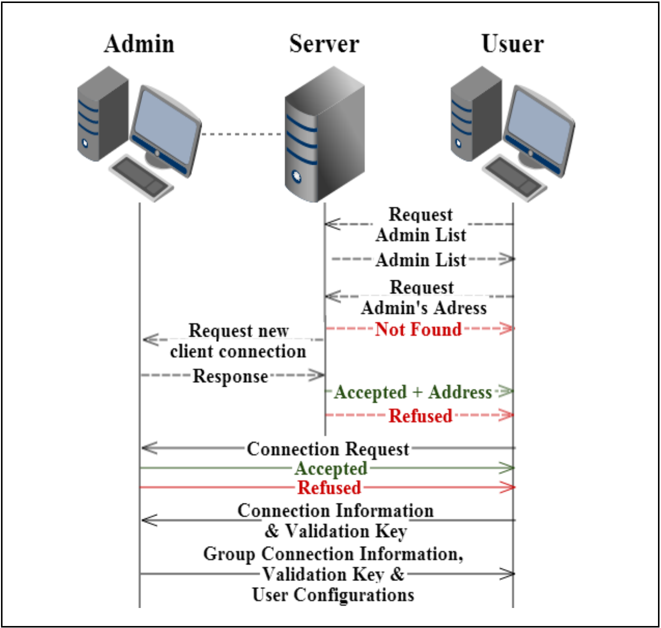

Figure 4. Connection Process.

The Admin must register its address in a server, so that common users can find him. Then, the Admin waits for connections from common users. The common user can request to server a list of registered Administrators and make a request of an address of an specific Admin. When the server receives the request, he sends a message to this

Admin requesting permission for the user to establish connection. If the user is accepted by Admin, the server sends to the user the admin address. Those messages are sent using UDP.

Once that user has the IP address and access to the Admin, he can realize the connection process. He sends a connection request over TCP to the admin, if accepted will be sent to user his UDP address, the group ID, a list with the connections information of the user from in the group, a set of flags that defines configuration and permission settings, and a two bytes validation key.

Subsequently, the address of the user is sent to the users connected to the group to which he was assigned. At the end of this process the user is added to the list of connected peers.

A. *Network Configuration Flags*

The network configuration flags describe the types of messages that will include the Admin as addressee and the permissions of each peer. Those flags' values are defined by the Admin during the connection process and are distributed over a Byte where each bit is equivalent to a Boolean that corresponds to a specific type of message. As shown in Table IV, the first four bits are related to common messages and the following four bits are related to streams. Since the control messages are always between an Admin and a common user there is no need to configure them.

TABLE IV. CONFIGURATION FLAGS

Type	1st bit	2nd bit	3rd bit	4th bit
Common	User	Group	Broadcast	System
Stream	User	Group	Broadcast	System

The bits corresponding to group messages and user, identify if those messages should include the Admin as addressee and the bits corresponding to Broadcast. The configurations of every user are saved by the Admin for validation purposes. To change the flags of a user, the Admin can send a control message with the new configurations and permissions.

V. CONCLUSION AND FUTURE WORK

In this work, we proposed an architecture model applied to a manageable P2P network that gives support to the development of multiplayer games that need to manage their peers connections and permissions. The users can communicate directly with each other, without the need of a server that would increase the costs of this process.

We presented a brief study of the NAT Transversal and some of the available techniques to break the NAT barrier in order to allow connections between hosts in different private networks. Were discussed the network model and the protocol that provides the functionalities that help both in the development of multiplayer games, as in the control of the network and in the managing of the connection processes.

As future work, we are developing a library, from the proposed architecture, in order to test the quality, usability and performance of developed applications.

REFERENCES

[1] M. Barton and B. Loguidici, "The History of Spacewar!: the best waste of time in the history of the universe," 2015 [On line]. Available from: http://www.gamasutra.com/view/ feature/132438/ the history_of_spacewar_the_best.php [retrieved: Mar., 2015].

[2] B. Yang. and H. Garcia-Molina, "Designing a Super-Peer Network," Proc. International Conference on Data Engineering (wICDE), Mar. 2003, pp. 49-60, ISSN: 1063-6382.

[3] Z. Cao, K. Li, and Y. Liu, " A Multi-Level Super Peer Based P2P Architecture," Proc. International Conference on Information Networking (ICOIN), Jan. 2008, pp. 1-5, ISSN 1617-5468, ISBN 3-88579-366-0.

[4] J.F. Kurose and K.W.Ross, Computer Networking: a top-down approach, Pearson Education, 6th ed., Mar. 2012, 864 p., ISBN-13: 978-0132856201, ISBN-10: 0132856204,

[5] S. Cheshire, M. Krochmal and K. Sekar, 2006. NAT Port Mapping Protocol (NAT-PMP). [Online] Internet Draft. Available from: http://tools.ietf.org/id/draft-cheshire-nat-pmp-02.txt [retrieved: Mar., 2015].

[6] RFC 3489, 2003. STUN – Simple Transversal of User Datagram Protocol [online] RFC. Available from: http://tools.ietf.org/html/rfc3489 [retrieved: Mar., 2015].

[7] RFC 5389, 2008. Session Transversal Utilities for NAT (STUN) [Online] RFC. Available from: http://tools.ietf.org/html/rfc5389 [retrieved: Mar., 2015].

[8] RFC 6886, 2013. NAT Port Mapping Protocol (NAT-PMP) [online] RFC. Available from: http://tools.ietf.org/html/rfc6886 [retrieved: Mar., 2015].

[9] RFC 6887, 2013. Port Control Protocol (PCP) [Online] RFC. Available from: http://tools.ietf.org/html/rfc6887 [retrieved: Mar., 2015].

[10] H. Suzuki, Y. Goto, and A. Watanabe, "External Dynamic Mapping Method for NAT Transversal, " Proc. International Symposium on Communications and Information Technologies, Octo. 2007, pp. 723-728, ISBN: 978-1-4244-0977-8.

[11] UPnP Forum, 2001. Internet Gateway Device (IGD) V 1.0 [Online] UPnP Forum. Available from: http://upnp.org/specs/gw/igd1 [retrieved: Mar., 2015].

[12] UPnP Forum , 2010. Internet Gateway Device (IGD) V 2.0 [Online] UPnP Forum. Available from: http://upnp.org/specs/gw/igd2 [retrieved: Mar., 2015].

[13] RFC 768, 1980. User Datagram Protocol [Online] RFC. Available from: http://tools.ietf.org/html/rfc768 [retrieved: Marc., 2015].

[14] RFC 793, 1981. Transmission Control Protocol [Online] RFC. Available from: http://tools.ietf.org/html/rfc793 [retrieved: Mar., 2015].

[15] A. S. Tanenbaum, Computer Networks, Editora Campus, 3rd ed., 1997.