PicoLAN-A Concurrent Multi-User Control System

Jayvijay V Shah, Vedant Ganguly, Krishna Patel, Prof. Chitra Bhole
K. J. Somaiya Institute of Engineering & Information Technology, Sion
Department Of Computer Engineering
Mumbai, India
jayvijay.s@somaiya.edu, vedant.g@somaiya.edu, krishna.patel@somaiya.edu, cbhole@somaiya.edu

***Abstract*—**PicoLAN is an application that controls multiple computers simultaneously in a local area network (LAN) using a single computer. The former machines are the slaves and the latter is the master. The master must execute commands such that they are executed on the slave nodes at the same time. PicoLAN is developed by using and extending the existing remote desktop sharing technologies. Since the number of existing solutions that support simultaneous remote control are very few and do not scale to an expected level, a genuine approach is made to develop a better substitute to the existing solutions using least resources. The concept of PicoLAN differs from that of remote desktop sharing where the increasing number of connected clients degrades the performance of the whole system which eventually makes the system unsuitable for remote collaboration in a network. PicoLAN is a system which will make simultaneous remote collaboration suitable also with a high number of connected slaves. All the peers form a group and collaborate to perform tasks like document-editing, web page sharing, demonstrating tutorials to a group of people with the help of remote control, software updating on multiple systems, software installation on multiple systems etc. PicoLAN supports unicast as well as multicast communications and it will prove to be of great helpto all those who want to connect and collaborate more effectively.

Keywords*-*LAN, UDP, TCP, multicast, robot, piconet.

*****

I. INTRODUCTION

In today's world, working in a team is very important in every sector like education, corporate world, offices etc. Working in a team requires computers to communicate with each other. The increasing number of ways in which computers communicate with each other led to the growth of LAN technology. LAN can be used as a medium which interconnects computers within offices, laboratory, school or any other building. Using LAN, communication between machines is easier and faster, even file transfers can be performed at a lightning fast speed using LAN.

Desktop sharing[1,4] has been one of the most trending fields in the technical world. Using desktop sharing it is possible for a group of people to work in collaboration. It allows people staying far away, to communicate with each other. It uses remote login wherein a user can connect to their desktop from any place. There are solutions that support real-time collaboration[2] but most of them can handle only a single computer. In other words, they are limited to one-to-one communication. But sometimes, it requires to handle more than 1 computer(s) at the same time. Controlling multiple computers concurrently becomes easy and fast with LAN technology.

Existing technologies can be brought into use to develop a concurrent real-time control system. For example, in piconet[6] (bluetooth), there are number of slaves and one master, utilizing the same physical channel. Here, the slaves need to synchronize their internal clock according to the hopping pattern of the master. In other words, the slaves follow the master. Such a type of behavior is incorporated in our application to achieve the desired outcome. Since we have used LAN as a physical channel, we have coined our application as PicoLAN. Using PicoLAN, the master (server) can control the slaves with the help of a single helper (client). Here the server, with the aid of the helper can simulate desktop sharing and communicate with each other. Later the server can include the slaves in its serving list, so that it can control multiple slaves as well as the helper by sending its mouse and keyboard events to them by multi-casting[3,5]. This transfers the server's events to all the connected machines simultaneously. In other words, server events are replicated and transferred to multiple connected slaves. The helper and the slaves identify the type of event sent by the server and execute them on their machines consuming their own resources. However, it is very interesting to see how the machines behave in such an environment. Such an application is of great help when multiple computers need to do the same work at the same time such as installing or uninstalling a particular software on all the machines or making a particular file available on all of them. It has been found that the number of naïve approaches that support concurrent muti-user control is limited. Simultaneous multiple interactions require a special treatment.

Our entire application is based on two things:

1. Firstly the quality of the network through which the computers are connected must be superior.
2. The machine which will act as a server must possess an ability to handle all the connected slaves in terms of its configuration.

In computer networks, a socket[8] is an end-point which acts as an interface for communication between multiple systems. The notion of a socket allows a single computer to serve many different clients at once. By the introduction of a port[10], which is a numbered socket on a particular machine, different types of information can be served. PicoLAN is developed using socket programming for connection between the participating machines and multicasting for sending commands to the connected slaves.

II. OVERVIEW

Lots of applications had been developed in the recent decade supporting remote desktop sharing, file transfer, chat, meeting, remote access etc., most of them allow one-to-one communication between machines, even few of them have introduced web based desktop sharing idea and implemented it efficiently as well. These existing systems utilize the screenshot of their server machine as a vital part of their system. The screenshot at the client's machine is just the replica of the server's desktop; this means that the whole system is dependent on the performance of the server. The client will not do anything else but it will just view what the server transfers to it. But a question arises that what if we want our connected clients to do the same work as the server does, but on their own (client) machines. In other words, we want our clients to mimic the server's actions. For e.g. if a user opens a browser on one machine, types a URL, then the same tab should get opened on rest of the computers simultaneously and automatically. Another instance is text-editing; when a user opens a word file on one machine and then it should get opened on all other machines. Editing the file from the main machine will edit the files of other machines perfunctorily. The same principle applies for closing an opened file. These examples are mainly static. That is, simply opening and closing a tab may not cause any change on a machine, hence it's static. The scope should not be limited to static events, but should also cover dynamic events. Consider a case in which we launch terminal on all the machines. This is a static action. Now type some command, say to install some software. This should automatically type the command on all the machines and in response should install it on all the machines. Thus it should allow us to perform dynamic events also, saving our time and effort. But while serving this purpose, the connected machines should not exhibit any sort of latency. Such a system is complex to implement; but would be of great help, if implemented.

In PicoLAN we have imitated the actions taking place on one computer on multiple computers at the same time, without any lag. Many applications also allow one to many communications, but there are some factors that degrade the performance of these systems. Here also the performance of our system is completely dependent on the server's ability.
To control multiple computers, we have used client-server architecture. Here the server, with the help of the client can simulate desktop sharing so that it is able to control the client and send various mouse and keyboard events to it. Now the slaves can connect to the master (server), such that they are able to receive the commands that the server sends to the client. This allows the commands to be executed on all the machines simultaneously.

Our application is entirely implemented on Java framework, using latest APIs such as Robot[7]. Java.awt.Robot class is used to take the control of mouse and keyboard. It is used to send server events like mouse move, key press, and key release with the help of java code. A socket object is used to connect the client and the server. Once we get the control, we can do any type of operation related to mouse and keyboard through our java code. The server waits for clients connections. When we move the mouse over the server side, this results in moving the mouse at the client side. The same happens when we right/left click a mouse button or type a key. Client receives server commands and executes them in the client PC. There are certain criteria that must be satisfied for PicoLAN to work effectively. First, all the machines must have the same version of operating system. The user-interface of the operating system must be uniform; for e.g. if desktop icon of firefox browser is located at position (x,y) coordinates in the master, it must also be the same for the slaves. One more criterion is the screen resolution, since we transfer the mouse events with their coordinates. So if screen resolution is not the same, this would hinder the proper functioning of the application.

III. METHODOLOGY

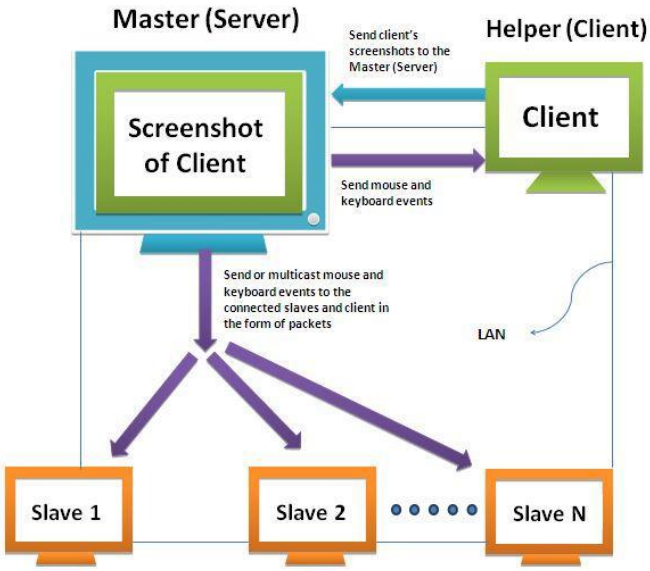

Fig. 1. Architecture of PicoLAN

As shown in Fig.1, we have used 3 modules to implement PicoLAN, viz. Master, Helper and Slave(s).

1) Master module: The master acts as the server which would control multiple clients at the same time. The server and client nodes are connected using TCP[9] to form client-server architecture. To be able to control the client from the server side, the server needs a replica of the client's screen. For this reason, a client first connects to the server. Then it will capture its screenshot and send it to the server, periodically. A window is used at the server side for displaying the client's screen. The purpose for using the client's screenshot on the server machine is to support interoperability between the server's machine and the client's machine. The master can quickly switch to its machine by just minimizing the screenshot window. The application need not to be ended and started repeatedly. This approach uplifts the overall efficiency and flexibility of the application. Now the client's screenshots simulate as a

client-screen on the server-side, so that the server can start executing mouse and keyboard events on it.

2) Helper module: The helper is the client machine which will send its screenshots to the server. After viewing the client-screen on its desktop, the server executes events like mouse-press, mouse-release, key-press and key-release. These commands are then transferred over the network to the helper and executed there itself. Now the master (server) is able to control the helper (client) resulting in a one-to-one communication or unicast.

3) Slave(s) module: To control more than one client (slaves), that is for one-to-many communication, multicasting is used. In computer networks, multicasting allows one-to-many communication between a single sender and multiple receivers. The remaining clients or the slaves can join a multicast group and connect to the Master. After the connection is setup, the next step is to control all the slaves at the same time. To achieve this, the mouse and keyboard events from the server-side are transferred in the form of Datagram packets over the network using UDP[9]. The slaves receive these commands in the form of packets and execute them using their own resources. In this manner, the master performs the action and the slaves mimic them.

IV. RESULT

The entire front end of PicoLAN is designed using NetBeans which is a software development platform that provides an integrated environment to develop softwares.

1)This is the master module. Out of the 3 versions, master is the first that needs to be initiated prior to the remaining 2 versions. Then the helper, and finally the slaves.

The master has to enter a port number and hit on Start Connection to connect with the helper. When a user clicks on "Start Connection", a directory is created at a static location. The purpose of this directory is to store all the files that are frequently needed by the user during aor subsequent session of PicoLAN. So that next time the user executes PicoLAN, he will find all the files that are to be used during that session at a single location i.e. at the PicoLAN directory. Clicking on "Empty PicoLAN Directory" button will delete all the contents of PicoLAN directory. Note that only the contents will be deleted, not the directory.

The user can end the session only by clicking on the "End Connection" button.

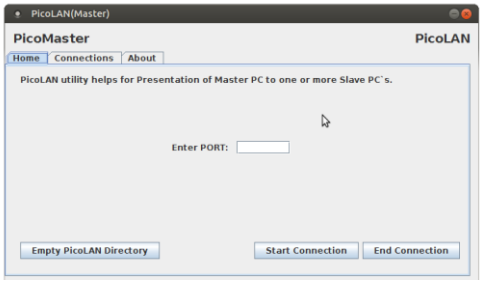

Fig. 2. PicoLAN (Master)

2) The helper must enter the IP address and port number same as the master has entered and then hit on Connect to Master. Now the screenshots of the helper will be sent to the master periodically. It should be noted that the server is the one who initiates the communication, so it should be opened for the helper to connect to it. Otherwise, the helper will continuously send the connection request to the server until the server receives it.

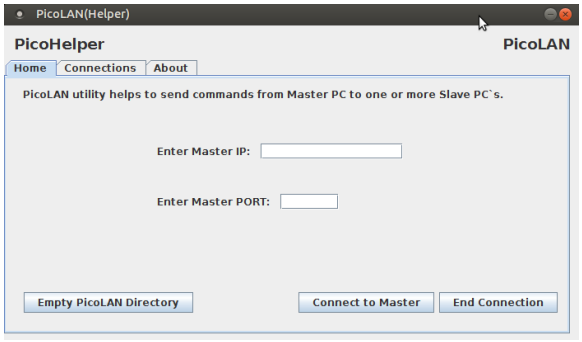

Fig. 3. PicoLAN (Helper)

3) This is the slave module. The slaves must hit on Start Connection to join the multicast group created by the master and connect with the it to receive commands.

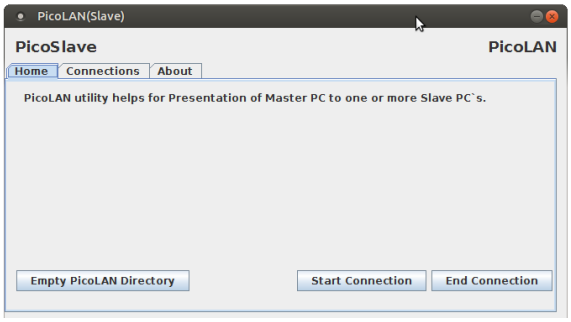

Fig. 4. PicoLAN (Slave)

V. CONCLUSION

PicoLAN helps to minimize the efforts needed to perform tasks that are required to be done repeatedly on different machines in a network by making a decent use of the LAN technology. This works smoothly as long as the different machines have the same version of the operating system installed. It also requires that the screen resolution of all the machines must be the same. The scope of the application is such that it allows to perform prime actions like opening a new word document, launching a browser, installing software's, closing an opened window, etc. However additional features such as file transfer, operating system independence and GUI independence can be incorporated in newer versions of the software.

REFERENCES

[1] Martin Prokes and RadoslavFasuga, "Tool for Desktop Sharing and Remote Teaching ForceB ", Emerging eLearning Technologies & Applications (ICETA), 2012 IEEE 10th International Conference, ISBN- 978-1-4673-5120-1 pp. 99 – 103, 2012.

[2] Tae-Ho Lee, Hong-Chang Lee, Jung-Hyun Kim, Myung-Joon Lee, "Extending VNC for Effective Collaboration", Strategic Technologies, 2008. IFOST 2008.Third International Forum, pp. 343 – 346, 2008.
[3] M. Hasan S. M. ; Lewis, G. J. ; Alexandrov, V. N. ; Dove, M. T. and Tucker, M. G.:, "Multicast Application Sharing Tool for the Access Grid Toolkit", UK e-Science All Hands Meeting, Nottingham, UK, 2005.
[4] Omer Boyaci, Henning Schulzrinne**,** "Application and Desktop Sharing", Proceeding CoNEXT '07 Proceedings of the 2007 ACM CoNEXT conference,
[5] Lawton, G., "Multicasting: will it transform the Internet?", Computer, vol:31 , Issue:7 pp: 13 – 15, 1998.
[6] Lafon, C, Durrani, T.S.,"New Bluetooth inter-piconet schedule with a slave to slave piconet formation". In: Personal Mobile Communications Conference, 2003, ISBN: 0-85296-753-5, pp: 81-85.
[7] Oracle JAVA Documentation Website http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html Retrieved: 08, 2015.
[8] Jitbanyud, A.&Toadithep, N, "The System of Powerful Computer Laboratory Class via Socket Programming", Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference, Vol:4, pp. 638 – 641, 2010.
[9] Oracle JAVA Documentation Website https://docs.oracle.com/javase/tutorial/networking/overview/networking.html
[10] Herbert Schildt,"Networking,"in*Java 2:The Complete Reference*,Fifth Edition, Tata McGraw-Hill Edition, 2002, ch.18, sec.partII,pp.589.