IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727Volume 6, Issue 5 (Nov. - Dec. 2012), PP 16-20
www.iosrjournals.org

Virtual Network Computing Based Droid desktop

[1,] Vaidehi Murarka, Sneha Mehta,[2]Dishant Upadhyay, [3]Abhijit Lal
[1, 2, 3]*Birla Institute of Technology and Science-Pilani, Hyderabad*

Abstract:*Our project is an Android application based on Virtual Network Computing" which helps people to access their laptops, personal computers using smartphones over a wireless network. Features like wireless data transfer between the android device and the client are also supported. The Integrated development environment used was Eclipse with the android plugin, and the server side (your laptop,pc etc.) programming was done using java.*

Client -server architecture was established for two way communication purpose where client was the android Smartphone and server was the windows desktop which communicate over TCP layer. The android client takes inputs from the users and sends the protocols for the execution of the mouse, keyboard and zooming events. The server parses such protocols and executes them using robot class of java. The desktop in turn takes screenshots continuously using robot class and sends them to the smartphone, thus establishing a two way communication network. The server sends the user interface to the client using the remote frame buffer. Remote frame buffer is the protocol used in virtual network computing. At the same time server receives commands from the client in the form of protocols. Since both the processes need to be performed simultaneously two threads are created, one to receive protocols from the client and the other to send user interface to the client.
In the similar way the client receives the screenshots in the child thread and sends the protocols in the main activity thread which gives the client a realistic environment of the Desktop. Threads need not be synchronized.
We created a simple Desktop server captures the screenshots of the machine it is running on and sends the images continually through an outputstream to the client in a particular string format. The server sends 30 screenshots per second to the client for the latter to obtain a live feed of the Desktop. The server waits for the input data stream from the client.

In order to completely access a desktop we need to access its hardware. However, the hardware cannot be directly accessed. So the android generates a standard protocol for each user interaction with the smartphones and sends it to the server. The touch events are considered analogous to mouse events and an onscreen keyboard of android replaces the standard IBM PC keyboard. Whenever data stream is sent by the client in the form of a string the server parses that string in accordance to our protocols and executes using robot class of java.

Some features like zooming and panning are also provided.

Keywords:*Protocol, Virtual NetworkComputing, Screenshots, Client-Server, Zooming-Panning, Mouse Driver*

I. Introduction

The network computer (NC) aims to give users access to centralized resources from simple, inexpensive devices. These devices act as clients to more powerful server machines that are connected to the network and provide applications, data, and storage for a user's preferences and personal customizations. This idea is taken to a further stage in virtual network computing (VNC). In VNC system, server machines supply not only applications and data but also an entire desktop environment that can be accessed from any Internet-connected machine using a simple software NC. Whenever and wherever a VNC desktop is accessed, its state and configuration (right down to the position of the cursor) are exactly the same as when it was last accessed.
In contrast to many recent Internet applications, which have focused on giving users access to resources located anywhere in the world from their home computing environments, VNC providesaccess to home computing environments from anywhere in the world. In Droid Desktop we have used the same concept of VNC and developed our own VNC server and client (which can be downloaded as an android app) so that a remote desktop computer can be accessed and worked with from an android smart phone or any other android devices.

II. Methodology:

1. Protocol generation

- ***Keyboard Protocol class***:

Whenever a key event is generated on the droid keyboard, android "onClick" method is called (by default). Using certain parameters the "onCLick" method distinguishes between the two layouts of the keyboard namely "alphapad" and "numpad". The alphapad consists of all the alphabets along with the facility of generating uppercase letters. The numpad consists of all the numeric values as well as the symbols, as on the

standard IBM PC keyboard. Modifier keys – ctrl, shift, alt, Navigation keys - Home, end, Editing keys – enter, delete, backspace, insert, spacebar, Miscellaneous keys – printscreen, esc are included in both the layout described above. The corresponding button ID is then sent to a class called "KeyboardProtocol". According to this ID Standard protocol is generated every time using a method "generateProtocol" in class.

- **Mouse Protocol**

Ifthe mouse click event is generated the following methods are calledonSingleClickListener,onLongClickListner (included in an interface onGestureListener),onDoubleTapListener(from an interface onDoubleTapListener).

The acknowledgement is made based in the activity performed. Single tap is used as mouse left click, double tap is used as mouse double click and long press is used as mouse right click. Once a gesture is generated these methods takes the counter to the class "MouseProtocol". This class generates the protocol using the method "generateProtocol".

- **Scroll Event Protocol**

Our application also interprets the scrolling event. There is a method called "onFling" built with an interface onGestureListener which is responsible for scrolling events. If this event is generated onFlingmethod sends the acknowledgement along with the direction of the fling to the class MouseProtocol, which generates the respective protocol.

III. Protocol

Client side architecture

The protocol generated in this application is a string of type "*********#",where "#" represents the end of the protocol .The first character helps server to determine whether protocol has been generated due to a mouse event , a keyboard eventor a scroll event.

The protocol made by the "generateprotocol" method for the different events is as follows-

- **Keyboard Protocol-**

Starting character-1

Protocol Format-1BUTTONID# where BUTTONID represents key (ASCII value) for the server to map corresponding button on the Desktop's keyboard.

Generated when the buttons corresponding to the Droid keyboard are accessed.

- **Scroll Event-**

Starting character-2

Protocol format-2SCROLLDIR# where SCROLLDIR is a string representing the direction of the scroll. The value of SCROLLDIR is "BOTTOM2TOP" and "TOP2BOTTOM" to scroll up and down respectively.

Generated by fling motion event.

- **Mouse Event-**

Starting character-3

Protocol Format-

3SLCLICK,x,y# →for single left click

3SRCLICK,x,y# -->for single right click

3DLCLICK,x,y# -->for double left click

Where x and y represent the coordinates of the layout where the touch event has occurred.

Generated by single tap, double tap and long press motion events respectively.

The protocol generated by any of the above events is immediately sent to the server through the data output stream.

Server side architecture

The server waits for the input data stream from the client. Whenever data stream is send by the client the server reads the data in pieces i.e. one protocol at the time by using # as a delimiter.

The server then understands the protocol generating event by reading the first character. The protocol received by the server is then parsed depending on the event that generated the protocol. The server knows that it needs to extract the direction of the scroll in case of scroll event protocol, the Ascii value of the button in case of keyboard event protocol and the type of click along with x ,y coordinates in case of mouse event protocol.

After the corresponding information has been extracted from the parsed protocol the server starts executing the protocols. Each event is executed using the java abstract windows toolkit robot class.

For executing the keyboard protocols, the key values extracted are passed to a switch statement in a class named as "Keyboard". The appropriate switch case calls Keyeventclass(java.awt.event.KeyEvent) which in turn returns the keycode of the physical key. This keycode is then passed to the keypress and keyrelease methods of the robot class that programmatically press and release the desktop keyboard keys.

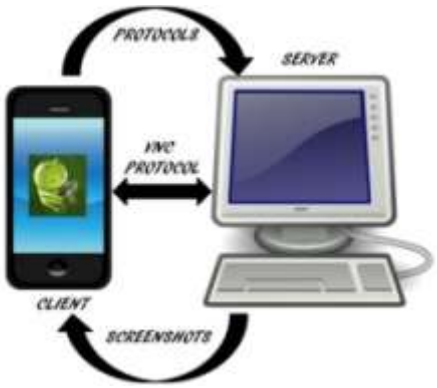
For the execution of the mouse protocol, the coordinates extracted from the protocol need to be scaled to the size of the Desktop's screen. This is done by simple interpolation (divide the coordinates by the mobile layout size and multiply by desktop screen size) using java AWT Dimension and Toolkit classes. Move the mouse cursor to the above calculated position using mousemove method of the robot class and press the mouse buttons by passing suitable parameters to the mousePress and mouseRelease methods of the robot class.

Scroll protocol can be executed using mouseWheel method in robot class. MouseWheel requires an integer as a parameter. The value of the parameter is +1 if the string extracted from the scroll event protocol is "BOTTOM2UP" and the value is -1 if the string is "TOP2BOTTOM".

IV. Communication Between Client And Server

The communication is established between client and server over the wireless network employing socket identified by an IP address concatenated with a unique port number. The desktop server waits for the incoming client request by listening to the specified port. Once the request is received, the server accepts a connection from the client socket to complete the connection over HTTP service.

Once the communication is established between the client and server data can be shared between them. But this is a two way communication process.



CLIENT-SERVER ARCHITECTURE

The server sends the user interface to the client using the remote frame buffer. Remote frame buffer is the protocol used in virtual network computing. At the same time server receives commands from the client in the form of protocols.Since both the processes need to be performed

Simultaneously two threads are created, one to receive protocols from the client and the other to send user interface to the client.

We created a class connect.java which implements the runnable interface creating a thread that continuously receives protocols from the clients. Another class Screeen_to_android.java creates a thread that sends the screenshots of the Desktop to the client.

In the similar way the client receives the screenshots in the child thread and sends the protocols in the main activity thread which gives the client a realistic environment of the Desktop. Threads need not be synchronised.

V. Screen Capture And Showing The Desktop On The Device

Server Side:

We created a simple server which accepts the request for connection from an android client, captures the screenshots of the machine it is running on and sends the images continually through an outputstream to the client in a particular string format.

The Robot class of java is used to create the screen capture.

The Algorithm for sending the screenshots while the server is still open is as follows:

While the server is connected to the client do the following:

- Get a new outputStream object.
- Get a desktop screenshot using robot.createScreenCapture(captureSize) method.

- Store the image into an array of bytes.
- Create a typical string str= "image" + dim.width + "x" + dim.height+ " " + (byteArray.length + " img=" + i + "\n"; where dim is the dimension of the screenshot.
- Convert the string into bytes by str.getBytes() method.
- Write the string and the previously stored image into an array of bytes to the output stream

Client Side:

A handler object is created in the main thread which then gets attached to its message queue.

```
Handler handler = new Handler() {
public voidhandleMessage(android.os.Messagemsg) {
Log.d("TEST", "handle mesasage called.");
display();
};};
```

An imageView is also created to display the changing images.
A global variable buffImgis created which contains the currently read image for the input stream.Adisplay () method is written in the main thread to display buffImg.

The client side consists of two separate threads. The main thread or the UI thread and the thread that accepts the desktop screenshots sent by the server (the background thread). In the background thread we create an input stream and accept images from the server one-by-one in the special string format in a loop. That is, in the loop we call the getString() method once, which accepts the string which consists of image its size(width * height),serial no. The string is then parsed to the location of the image and the image is read from the inputstream in the form of bytes. The image is now decoded into bitmap format and stored in buffImgand a message is passed to the handler (attached to the main thread).
In the main thread, in the message queue the message arrives,which contains a call to the display function which displays the buffImg.
This activity continues in loop unless the all the image sent by a server are read.

VI. Mouse Driver

A driver to track the motion of the cursor was also implemented as cursor is not a part of the remote frame buffer in windows.

VII. Zooming And Panning

To provide a more clear view of the desktop on the smartphone screen zooming and panning facility was also provided.Multi-touch zoom facility was implemented using "ScaleGestureDetector" class in android and was based on the concept of matrices.

VIII. Image Compression

The size of the screenshots sent by the server to the smartphone is 1366*768 pixels, making the image too heavy for the smartphone to handle. Since the communication is wireless,the larger is the size of the screenshot, more is the lag in the display of the Desktop on the smartphone. Therefore size of the screenshots needs to be reduced by the server using various image compression techniques. Two things have to be observed when image compression is to be done- delay in the display of the image should be minimized, at the same time image quality of the Desktop displayed on the smartphone should not be compromised during zooming and panning.

IX. Conclusion

Droiddesktop is an internet packet transferring application which is implemented on Android platform and nowadays Android is stepping into a next level of mobile internet. So, in the long run there is a strong objection that we can develop many useful apps based on Droiddesktop. And as far as the runtime constraint is concerned Droiddesktop is efficient because of synchronous threads. Hence, we conclude that Droiddesktop is an application with minimum amount of redundant data and a strong future because of its wireless features.

Acknowledgement

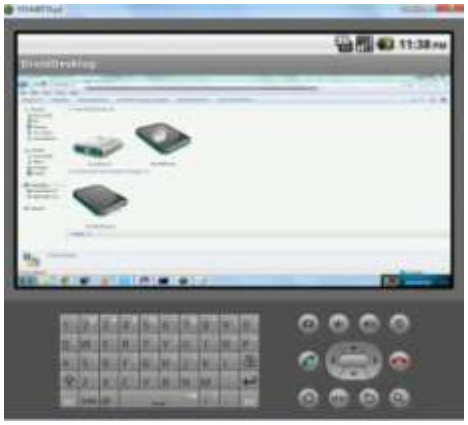
We would like to express our gratitude to the faculty at BITS PilaniHyderabad Campus for their support. We are especially grateful to Mr. AbhayRaghuwanshi and MrsGagandeepKaur from Global Logic, Nagpur for their help and advice in the project work.

References

[1] http://cursos.leon.uia.mx/temporal/software/vnc/windows/tr.98.1.pdf
[2] www.realvnc.com/docs/rfbproto.pdf
[3] www.realvnc.com/products/vnc/.../5.../VNCUser_Guide.pdf
[4] www.wikipedia.org/wiki/Virtual_Network_Computing
[5] www.wikipedia.org/wiki/RFB_protocol

X. Results and Discussions

The main advantage of our methodology is that it relies on the computational ability of the laptop and not of the smartphone. And the screen shots of the laptop are not stored in a file but are directly displayed onto the screen thus avoiding memory wastage. Snapshots of the working program are given below-



Screenshot of the Application Simulation on the Emulator Screenshot of Zooming and PanningFacility