International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Pp-30-34 ISSN: 2249-6645

Remote Desktop Access Using Remote Frame Buffer in Mobile Cloud Environment

Mrs. P.Visalakshi [1], M.Deepak[2]
[1]MCA., M.S.,(Ph.D) Asst. Professor, (S.G), Department of Computer Applications SRM University Kattankulathur,Chennai-603203
[2]Department of Computer Applications SRM University Kattankulathur, Chennai-603203

ABSTRACT: *Cloud computing is defined as the trend in which resources are provided to a local client on an on-demand basis, usually by means of the internet. Mobile Cloud Computing is the emerging area with the usage of Cloud Computing in combination with mobile devices. The principle of mobile cloud computing physically separates the user interface from the application logic. Although they suffer from intrinsic resource limitations, mobile devices have become very popular. Mobile cloud computing provides a solution to meet the increasing functionality demands of end-users, as all application logic is executed on distant servers and only user interface functionalities reside on the mobile device. The mobile device acts as a remote display, capturing user input and rendering the display updates received from the distant server .The major challenge for mobile cloud computing is the limitations inherent to the mobile devices like processing power, memory, network bandwidth and storage capacity as compared to a fixed device like PC, and short battery lifetime and interaction latency introduce another major challenges for the remote display of cloud applications on mobile devices. In this paper, a number of adequate solutions that have been proposed to tackle the main issues associated with the remote display of cloud services on mobile devices.*

Keywords: *Mobile Cloud, RFB Protocol, VNC server, PAAS.*

I. INTRODUCTION

Mobile cloud computing (MCC) [1], is a combination between mobile network and cloud computing, thereby providing optimal services for mobile users. In mobile cloud computing, mobile devices do not need a powerful configuration (e.g., CPU speed and memory capacity) since all the data and complicated computing modules can be processed in the clouds.

Mobile devices (e.g., smartphone, tablet pcs, etc.) are increasingly becoming an essential part of human life as the most effective and convenient communication tools not bounded by time and place. Mobile users accumulate rich experience of various services from mobile applications (e.g., iPhone apps, Google apps, etc.), which run on the devices and/or on remote servers via wireless networks. The rapid progress of Mobile Cloud Computing becomes a powerful trend in the development of IT technology as well as commerce and industry fields. However, the mobile devices are facing many challenges in their resources (e.g., battery life, storage, and bandwidth) and communications (e.g., mobility and security) [2]. The limited resources significantly impede the improvement of service qualities. MCC at its simplest refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device.

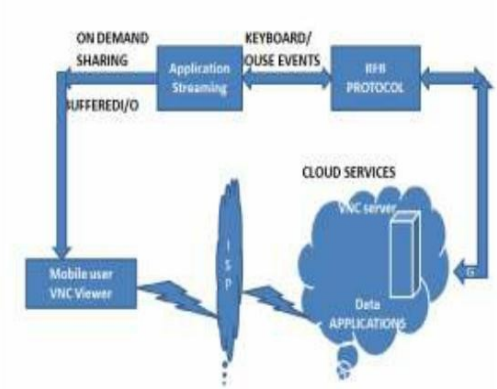


Fig.1 Architecture Diagram

Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smartphone users but a much broader range of mobile subscribers. MCC as a new paradigm for mobile applications whereby the data processing and storage are moved from the mobile device to powerful and centralized computing platforms located in clouds. These centralized applications are then accessed over the wireless connection based on a thin native client or web browser on the mobile devices.

II. ADVANTAGES

2.1 Application Streaming

This platform allows applications to be deployed in real-time to any client from a virtual application server. It removes the need for local installation of the applications. Instead, only the SoftGrid runtime needs to be installed on the client machines. All application data is permanently stored on the virtual application server. Whichever software is needed is streamed from the application server on demand and run locally.

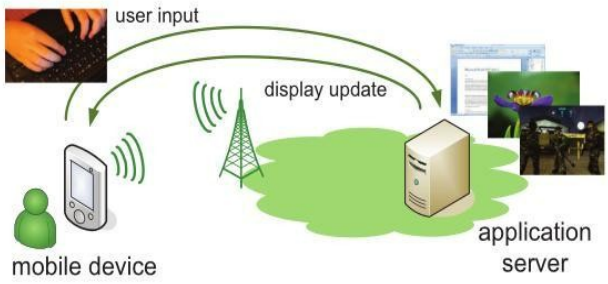


Fig2: The viewer component on the client forwards the captured user input to the server.

2.2 Downstream data peak reduction

Interactive applications only update their display unless instructed by the user. Usually, these display updates involve a large amount of data that needs to be sent to the client in a short interval to swiftly update the display Moving or capable of moving with great speed. Their analysis of remote display protocol traffic traces reveals a lot of redundancy, caused by the repainting of graphical objects after recurring user actions. They propose a hybrid cache-compression scheme whereby the cached data is used as history to better compress recurrent screen updates. The cache contains various drawing orders and bitmaps.

2.3 Optimization of upstream packetization overhead

Some network send in desecrate chunks to call packets User events are the principal source of remote display traffic in the upstream direction from client to server. Individually, each user event embodies only a small amount of information: a key or button id, one bit to discriminate between the press and release action and possibly the current pointer coordinates. Nevertheless, user events induce important upstream traffic because they are often generated shortly after each other.

Entering a single character results in two user events to indicate the press and release action, a large packetization overhead is observed owing to the headers added at the TCP, IP and (wireless) link layer. The upstream packetization overhead of three commonly used remote display protocols. The maximum buffering period is a consideration of remote display bandwidth reduction against interaction latency The highest bandwidth reductions are achieved for interactive applications with frequent user events and lower roundtrip times

III. RFB (REMOTE FRAME BUFFER PROTOCOL)

RFB ("Remote Frame Buffer") is a simple protocol for remote access to graphical user interfaces. Because it works at the frame buffer level it is applicable to all windowing systems and applications, including X11, Windows and Macintosh. RFB is the protocol used in VNC (Virtual Network Computing). The remote endpoint where the user sits (i.e. the display plus keyboard and/or pointer) is called the RFB client or viewer.

The endpoint where changes to the frame buffer originate (i.e. the windowing system and applications) is known as the RFB server. Although RFB started as a relatively simple protocol it has been enhanced with additional features (such as file transfers) and more sophisticated compression and security techniques as it has developed. To maintain seamless cross-compatibility between the many different VNC client and server implementations, the clients and servers negotiate a connection using the best RFB version, and the most appropriate compression and security options, that they can both support.

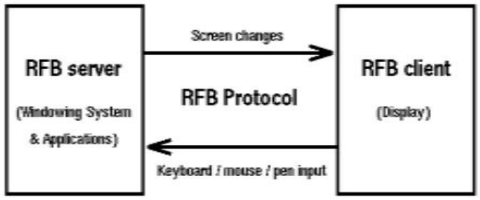


Fig.2: Remote Frame Buffer (RFB)

RFB was originally developed at Olivetti Research Laboratory (ORL) as a remote display technology to be used by a simple thin client with ATM connectivity called a Videotile. In order to keep the device as simple as possible, RFB was developed and used in preference to any of the existing remote display technologies. RFB found a second, and more enduring, use when VNC was developed. VNC was released as open source software and the RFB specification published on the web. Since then RFB has been a free protocol which anybody can use. When ORL was closed in 2002 some of the key

people behind VNC and RFB formed RealVNC Ltd in order to continue development of VNC and to maintain the RFB protocol. The current RFB protocol is published on the RealVNC website.

3.1 Display protocol

pixel formats are 24-bit or 16-bit "true color", where bit-fields within the pixel value translate directly to red, green and blue intensities and 8-bit "color map where an arbitrary mapping can be used to translate from pixel values to the RGB intensities. Encoding refers to how a rectangle of pixel data will be sent on the wire. The encoding types defined at present are Raw, CopyRect, RRE, Hextile and ZRLE. In practice we normally use only the ZRLE, Hextile and CopyRect encodings since they provide the best compression for typical desktops.

3.4 Protocol Messages

The display side of the protocol is based around a single graphics primitive: "put a rectangle of pixel data at a given x & y position". A sequence of these rectangles makes a frame buffer update (or simply update). An update represents a change from one valid frame buffer state to another. The rectangles in an update are usually disjoint but this is not necessarily the case. The update protocol is demand-driven by the client. That is, an update is only sent from the server to the client in response to an explicit request from the client. This gives the protocol an adaptive quality. The slower the client and the network are, the lower the rate of updates becomes. With typical applications, changes to the same area of the frame buffer tend to happen soon after one another. With a slow client and/or network, transient states of the frame buffer can be ignored, resulting in less network traffic and less drawing for the client.

There are three stages to the protocol. First is the Handshaking phase, the purpose of which is to agree upon the protocol version and the type of security to be used. The second stage is an Initialization phase, where the client and server exchange ClientInit and ServerInit messages. The final stage is the Normal protocol interaction, the client can send whichever messages it wants, and may receive messages from the server as a result. All these messages begin with a message-type byte, followed by any message-specific data. The following descriptions of protocol messages use the basic types U8, U16, U32,S8, S16, S32. These represent respectively 8, 16 and 32-bit unsigned integers and 8, 16 and 32-bit signed integers. All multiple byte integers (other than pixel values themselves) are in big endian order (most significant byte first).

3.2 Input protocol

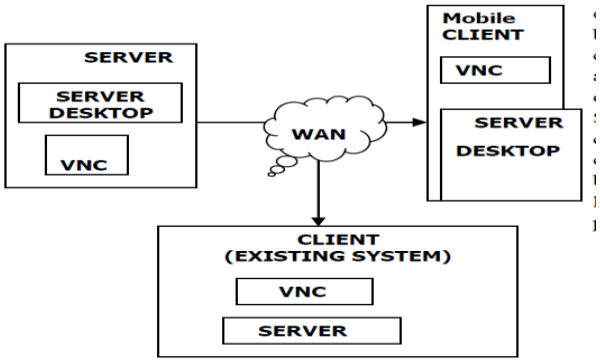
The input side of the protocol is based on a standard workstation model of a keyboard and multi-button pointing device. Input events are simply sent to the server by the client whenever the user presses a key or pointer button, or whenever the pointing device is moved. These input events can also be synthesized from other non-standard I/O devices. For example, a pen-based handwriting recognition engine might generate keyboard events.

3.3 Representation of pixel data

Initial interaction between the RFB client and server involves a negotiation of the format and encoding with which pixel data will be sent. This negotiation has been designed to make the job of the client as easy as possible. The bottom line is that the server must always be able to supply pixel data in the form the client wants. However if the client is able to cope equally with several different formats or encodings, it may choose one which is easier for the server to produce. The most common

IV. PROCESS METHODOLOGY

Step1: The VNC server PC is run, which is connected to the Internet.
Step 2: The mobile having GPRS is activated which is used for viewing other PCs, which is connected to VNC Server. In this case the PC will act as Server and mobile act as client. Connection happens through Wide Area Network.
Step3: Using J2ME wireless tool kit, the mobile checks PC, which is to be viewed, and it will check its IP address and password for authentication.
Step 4: After authentication if the IP address of the PC is valid, it will connected to the server and we can view the PC using the mobile and if it is not valid the connection will not happen.
Step 5: After connecting the PC to the server, all the operations of the PC can be performed using the mobile. For example, opening a file, deleting, cut, copy paste etc.



V. VNC Servers

Writing a VNC server is slightly harder than writing a client for a number of reasons. The protocol is designed to make the client as simple as possible, so it is usually up to the server to perform any necessary translations. For example, the server must provide pixel data in the format the client wants. We have servers for our two main platforms, X (i.e. Unix) and Windows NT/95. A Unix machine can run a number of Xvnc servers for different users, each of which represents a distinct VNC desktop. Each VNC desktop is like a virtual X display, with a root window on which several X applications can be displayed.

The Windows server (WinVNC) is a little more difficult to create, because there are fewer places to insert hooks into the system to monitor display updates, and a less clearly defined model of multi-user operation. Our current server simply mirrors the real display to a remote client, which means that the server is not 'multi-user'. It does, however, provide the primary user of a PC with remote access to their desktop. We have also created simple servers, which produce displays other than desktops, using a simple toolkit. A "VNC CD player", for example, generates a CD player user interface using the protocol directly without any reference to a windows system or frame buffer. Such servers can run on very simple hardware, and can be accessed from any of the standard viewers.

VI. VNC Clients

Writing a VNC viewer is a simple task, as it should be for any thin-client system. It requires only a reliable transport (usually TCP/IP), and a way of displaying pixels (either directly writing to the frame buffer, or going through a windowing system). We have clients for all the networked display devices we have available at our lab. This includes the Videotile (the original RFB client), an X-based client (which runs on Solaris, Linux and Digital Unix workstations), a Win32 client, which runs on Windows NT and 95, a Macintosh client, and a Java client, which runs on any Java-capable browser (including Sun's Java Station). Members of our lab use these clients on a daily basis to access their personal computing environments.

Fig.3: VNC Viewer

VII. CONCLUSION AND FUTURE WORK

By physically separating the user interface from the application logic, the principle of mobile cloud computing allows to access even the most demanding applications in the cloud from intrinsically resource-constrained mobile devices. In this paper, we have surveyed contemporary remote display optimization techniques specifically tailored to the short mobile device battery lifetime, the varying and limited bandwidth availability on wireless links and the interaction latency. Although each of these solutions adequately addresses specific challenges of mobile cloud computing, an overall approach is currently lacking. The context of mobile cloud computing is highly dynamic, owing to the user mobility, the wide diversity of applications, and the varying wireless channel status. Future work should therefore be devoted to the design of an overall framework, integrating all the presented solutions, and activating the most appropriate solutions dependent on the current device, network and cloud server status. We are extending our implementation to speed up the frame rate, to incorporate more intelligent navigation, to provide integrated panning and zooming of the view port, to simplify basic operations we apply speed-dependent automatic zooming, to support incremental updating of the VNC viewer image**.**

REFERENCES

[1] Guan, Le; Ke, Xu; Song, Meina; Song, Junde,"IEEE A Survey of Research on Mobile Cloud Computing" pp : 387 – 392, May 2011.
[2] Pieter Simoens, Filip De Turck member,, Bart Dhoedt member, Piet Demeester Fellow, "IEEE RemoteDisplay Solution for MobileCloud Computing" Vol: 44 ,pp : 46 - 53 ,Aug 2011.
[3] K.-J. Tan, J.-W. Gong, B.-T. Wu, D.-C. Chang, H.-Y. Li, Y.-M. Hsiao, Y.-C. Chen, S.-W. Lo, Y.-S. Chu, and J.-I. Guo, "A remote thin client system for real time multimedia streaming over VNC," in International Conference on Multimedia and Expo (ICME), 2010, pp. 992–7.
[4] K.Pentikousis,"In Search of Energy-Efficient Mobile Networking,"IEEE COMMUNICATIONS MAGAZINE, vol. 48, no. 1, pp. 95–103,JAN 2010.
[5] B.-G. Chun, P. Maniatis, Dynamically partitioning applications between weak devices and clouds, in: Proceedings of the 1st ACM Workshop on Mobile CloudComputing & Services: Social Networks and Beyond, MCS '10, ACM, 2010, pp. 7:1–7:5.
[6] Mahadev Satyanarayanan,"Mobile computing:The next decade,"Proc. 11th Intl. Conf. on Mobile Data Management (MDM'10), Kansas, MO, 2010.
[7] Cao,"Mobile community cloud computing: emerges and evolves," Proc. 1st Intl. Workshop on Mobile Cloud Computing (in conjunction with) 11th Intl. Conf. on Mobile Data Management (MDM'10), Kansas, MO, 2010, pp. 393-395.
[8] A. Klein, C. Mannweiler, J. Schneider, and D. Hans, "Access Schemes for Mobile Cloud Computing," in Proceedings of the 11th International Conference on Mobile Data Management (MDM), pp. 387, June 2010.
& G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in Proceedings of the 1st ACM Workshop on Mobile Cloud Computing Services: Social Networks and Beyond (MCS), no. 6, 2010.
[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Casefor VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Computing, vol. 8, no. 4, pp. 14–23, OCT-DEC 2009.
[11] "J2ME: The Complete Reference"- McGraw-Hill, James Keogh
[12] "J2ME in a nutshell"- O'Reilly March 2002, Kim Topley
[13] "Wireless J2ME Platform Programming"- SUN Microsystems, Vartan Piroumian.
[14] "Java 2 Micro Edition"- McGraw-Hill International 2002 Edition, Ortiz.
[15] "Wireless Java Programming with J2ME"-O'Reilly 2003, Chris Seguin.
[16] "Core J2ME technology and & MIDP" - SUN Microsystems, John W. Muchow.
[17] "Java 2 the complete reference"- McGraw-Hill fourth Editions Herbert Schildt.
[18] D. Kovachev, D. Renzel, R. Klamma, and Y.