Android Diskless System

Abel Alfonso Fírvida Donéstevez **email: aafirvida@uci.cu**,
Roberto Espí Muñoz, Cealys Alvarez Trujillo,
Miguel Albalat Aguila, Allan Pierra Fuentes.

Abstract—In this paper we propose the Android Diskless System, which provides a set of tools to boot and manage diskless computers on a LAN. The motive is to use old computers, still in good conditions but without a hard drive, for a general purpose environment such as school labs or small businesses like cyber cafes. The resulting solution allows users to connect to the Internet, store their data, play games, and interact with documents (pdf, doc, xls, ppt) and media, which is the common use of computer nowadays.

At the end of paper the reader will find benchmark tests applied to the system demostrating that there are no remarkable differences in performance between diskless and standard Android systems and also that other diskless GNU/Linux solutions aren't as efficient as ours for our purpose.

Index Terms—Android, diskless clients, obsolete computers.

I. INTRODUCTION

IN the 1960s, the expected market availability for specific computer chips was between 20 and 25 years; today it's between 2 and 5. Called the dark side of Moore's Law, which states that the number of transistors on a chip doubles every 18 to 20 months: poor planning for parts obsolescence causes companies to spend progressively more in dealing with the effects of aging systems–which leaves even less money for new investment, thus creating a downward spiral of maintenance costs and delayed upgrades[1].

This trend affects in a greater share the public sector of development countries that can not afford renewing their hardware when it becomes obsolete and must extend the useful life of computers as long as they are functional. A good option in solving this problem is the use of diskless clients using a Linux-based operating system (OS) since Linux developers claim that a diskless client using their OS could have a lifespan of more than 300 years without any hardware or software upgrades[2].
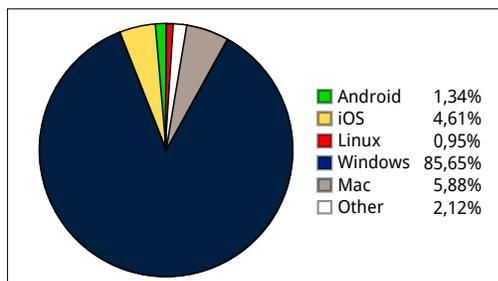

Fig. 1: OS market share[3], [4].

In order to select a Linux-based OS for diskless clients it is necessary to study the OS's market share (Figure 1), which brings us to the conclusion that the Android OS is more popular than all other Linux distributions joined together. Android was also designed to optimize the use of memory and hardware resources[5] and has a great number of free applications that could provide a good set of tools in everyday computing.

There are some projects, such as LTSP[6] or Zentyal[7], that provide tools to manage diskless computers and thin clients. However, to the best of our knowledge, there is still no project that allows Android to run on a computer without a hard drive. Another aim is to use only free and open source software to make the solution more suitable for developing countries which are the most affected by the problems of aging technologies.

This paper presents the design and implementation of a diskless computer network using the Android OS. Section II discusses projects and papers related to this investigation, Section III describes the basic design of our system, Section IV details a proof-of-concept prototype that we have implemented. Section V depicts tests results of our diskless system's benchmarking and Section VI concludes this paper.

II. RELATED WORK.

There are some projects that aim to provide a solution to the aging technologies issue such as the Lightweight X11 Desktop Environment (LXDE) which is an extremely fast-performing and energy-saving desktop environment, that can be installed on many Linux distributions including Debian, Fedora, OpenSUSE, Ubuntu and also runs on OpenSolaris and BSD[8]. LXDE has a good performance even on a personal computer produced back in 1999 but its design and information architecture is based on MS Windows 98/2000, a paradigm that has evolved in the last ten years and is getting obsolete with the new concepts of desktop enviroments that can be seen in Plasma from KDE, GNOME Shell, the Windows 8 Desktop and the launcher of Android 4.

The *Virtual Smartphone over IP* work of Eric Y. Chen and Mistutaka Itoh reveals a platform to empower smartphones with cloud computer processing in a model close to the thin clients architecture[9]. Even when the aim of that investigation if far from ours, there are some elements to take into consideration in their platform such as the use of Network File System (NFS) to provide the nodes of an external storage directory.

Also the paper *Implementation of a Diskless Cluster Computing Environment in a Computer Classroom* of Chao-Tung Yang, Wen-Feng Hsieh and Hung-Yen Chen was reviewed to select the server side technologies for our system.

III. System Overview.

The functioning of diskless computers on Linux based OS can be implemented in two ways: with the processing on the server side, as thin clients, or with the processing on the client side. Both of them share the same boot sequence (Figure 2).
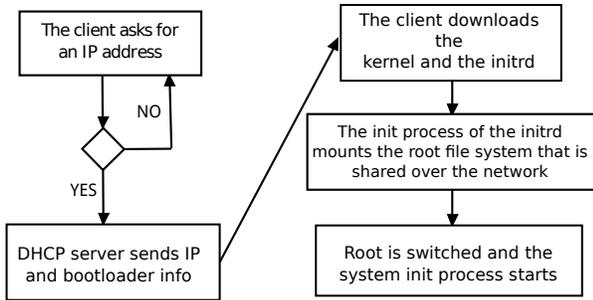

Fig. 2: Boot sequence of a diskless computer[1].

After the last step of booting, the client can connect to a server through any protocol (e.g. VNC, RDP, SSH or XDMCP) that allows running graphical applications remotely (server side processing) or it can login locally (client side processing).
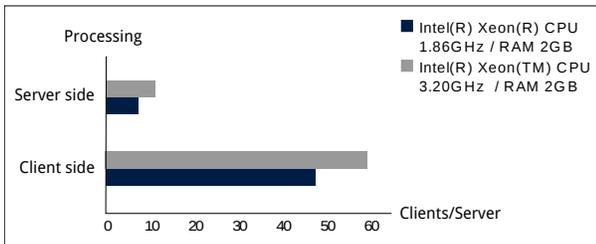

Fig. 3: Clients/Server test using GNU/Linux

For this investigation we chose the client side processing because it is less expensive since it supports a better ratio of clients per server without crashing (Figure 3). The proposed Android Diskless System architecture is just like any other thin client network implementation. It is composed of a number of diskless computers connected to a server or servers that provide:

- DHCP, which is the main service since it provides the clients with all the necessary information to boot using the network.
- Advanced Trivial File Transfer Protocol (ATFTP), to share the PXE boot-loader's configuration files, the Linux kernel and it's initial RAM disk (initrd). It was selected among all other TFTP servers since it supports a configurable amount of client connections.
- Network Block Devices (NBD). A service which presents a remote resource as local to the client [11]. Using it our system shares the root file system and a swap memory space to the clients.

---

[1]The Linux initial RAM disk (initrd) is a temporary root file system that is mounted during system boot to support the two-state boot process. The initrd contains various executables and drivers that permit the real root file system to be mounted, after which the initrd RAM disk is unmounted and its memory freed[10].

- NFS, This protocol is used by diskless nodes for all persistent file storage. It operates in the same manner as an SD card holds data for physical smart-phones. Since the NFS is easily scalable, it practically provides each client an unlimited file storage[9].

The solution can be easily scaled by separating the services one per server without making any changes to the software.

IV. Implementation.

A. Android modifications.

As a proof-of-concept to make the client's OS we used the first release candidate of the version 4.0.3 of Android x86 project[12] to which we added the modules to support NBD, NFS and SWAP memory space[2] in the Linux kernel layer (Figure 4).
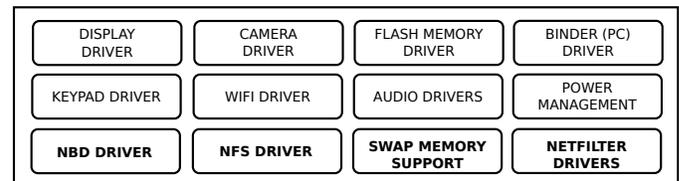

Fig. 4: Kernel layer of Android system architecture[14].

The init script of the initrd was modified to locate the root file system, the user data image and the swap space in the remote server. This script's functions are based on Busybox applets, so **nbd-client**, **swaponoff** and **nfs mount** support were included in Busybox.

```
# get IP
udhcpc -s /sbin/udhcpc.sh -i eth0

# create nbd devices
for i in 1 2 3; do
  mknod /dev/nbd$i b 43 $i;
done

# The variables $SERVER, $DATA_PORT
# and $SWAP_PORT are passed to
# the script via kernel command line
nbd-client "$SERVER" 9000 /dev/nbd0

if ! [[ -z "$DATA_PORT" ]];then
    nbd-client $SERVER $DATA_PORT /dev/nbd1
    export DATA=/dev/nbd1
fi

if ! [[ -z "$SWAP_PORT" ]];then
    nbd-client $SERVER $SWAP_PORT /dev/nbd2
    swapon /dev/nbd2
fi

# check the remote root file system
check_root /dev/nbd0
```

Sample Code 1: Initrd's init script modifications.

---

[2]Swap memory space provides a more stable foundation, particularly for production systems running programs that consume large amounts of memory[13] or for those computers that do not have enough RAM.

As shown in Sample Code 1 the root filesystem is exported to all clients throughout the same port (9000), the swap space and the user data image are shared among each node through different ports that are provided to the init process in the kernel's command line. As a result of this,

1)

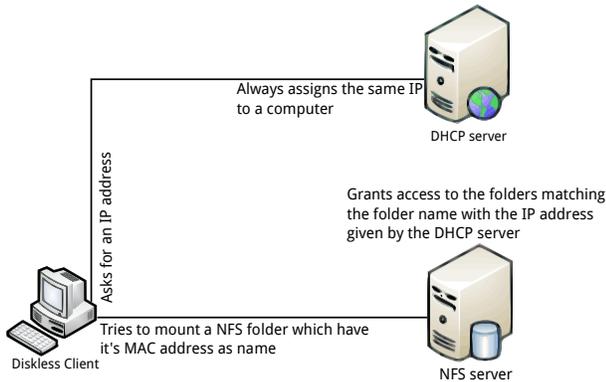clients share the same file system but never the user data or swap space.

Fig. 5: External storage mount process.

The init script is powered with a function that takes the MAC address of the network interface and the NFS server IP and mounts the NFS folder with the name MAC as the external storage directory. In order to ensure a first security level the DHCP server always gives the same IP address to the computers using the client's MAC as it's key and the NFS server grants access to the folder with the name "MAC" only to this IP. Therefore no single computer has access to the storage of another (Figure 5).
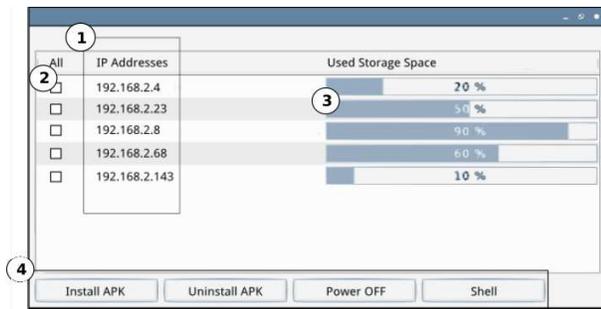
B. Clients administration tool.

Fig. 6: Administration's tool prototype interface.

We created a tool based on the Android Debug Bridge (ADB)[15] to manage all the diskless clients. This tool provides a way to easily manage applications, reboot and halt the remote system in all the clients at the same time or one by one.

Clients administration tool (Figure 6) components description:

1) IP addresses list for all the clients that are running.
2) Check-box to select the clients to which apply actions.
3) Storage space indicator for each client.
4) Action buttons.

The ADB tool can be used from any computer to administrate an Android OS, so our system includes security measures to make this service accessible only from the administration server. To ensure this we added netfilter support to the kernel layer (Figure 4) and the iptables applet to the Busybox program that runs on the Initrd. On the init process all connections to the ADB's port (5555) are rejected unless they come from the management server.

V. EVALUATION.

A. Benchmarking.

To demonstrate that the Android Diskless System could run flawlessly on very old computers turned into diskless nodes we used the tool **0xbench** from **0xlabs** which is an open source benchmarking suite with 16 tests included[16]. Results were compared with their counterparts on the same computers with a hard drive.

TABLE I: Benchmarking tests on ASUS P4P800.

	Diskless	Using a SEAGATE ATA ST3500418AS hard drive
SciMath	s	
Composite	14.0648279348993	14.0780899274729
Fast Fourier Transform	8.23625260632359	8.23874281784106
Jacobi Successive Over relaxation	25.483430443781	25.591729622668
Monte Carlo integration	3.16193276747629	3.19931657785062
Sparse matrix multiply	14.7126428415382	14.7126428415382
Dense LU matrix factorization	18.7298810153771	18.6480177774666
2D	fps	
Canvas Redraw	60.012127	60.084198
Draw Circle	58.91174	57.539078
Draw Rect	29.417336	31.284286
Draw Arc	52.293583	52.17073
Draw Text	59.745113	59.677807
Draw Image	15.1775	15.153122
3D		
GL Cube	59.56045	59.558086
GL Teapot	39.21261	39.21261
GL Blending	63.730804	63.593002
GL Fog	63.554623	63.60514

As shown in TABLE I there is no remarkable difference in the performance of diskless nodes compared to that of standard computers.

Other series of tests were applied to compare the performance of our system against a GNU/Linux distribution running without a hardrive. The OS customization to these test was done using Ubuntu 10.04 and LXDE as the desktop environment. Figures 7, 8 and 9 depict the results:

- CPU usage is almost the same in both OS, with maximun values of 51% for Android and 47% for LXDE and an average of 8.9% and 9.9% respectively.
- I/O output requests made by the OS are significant at boot time, reaching values of 10500Kb/s for our system and 2800Kb/s for the other, but once their boot processes are completed the average is insignificant for both approaches.
- Even when the difference between RAM used in both systems has an average of 24% the trend of Android memory consumption is to oscillate from 70% to 80% while the other system trends to increase progressively

which could take it in a longer experiment to exceed Android's consumption level.

VI. Conclusions and Further Work

There are currently 475843 applications in the Android market from which 27% are targeted as low quality applications, 348993 are free and 128343 are downloadable by payment[17], [18]. In the upcoming future there will be Android applications for all the purposes of a common user. The Android Diskless System discussed in this paper can be used in classroms, libraries, navigation points, kiosks or even in Arcade machines using those apps.

Our aproach is as good as any other GNU/Linux for diskless computers, but carries with it the new concepts of desktop enviroments, uses the most popular Linux based OS and provides tools to manage almost every setting of the diskless nodes.

The next step in this investigation will be to create custom roms for specific environments such as schools or libraries and test them in real envioronments to demostrate the factibility of the proyect. Meanwhile we will wait the release of Android 5 Jelly Bean, which will have native support for desktop computers, to integrate it to the Android Diskless solution.

References

[1] P. Standborn, "Trapped on technology's trailing edge," *IEEE Spectrum*, 2008.
[2] "Diskless nodes HOW-TO document for linux: Advantages of diskless computer," http://linuxdocs.org/HOWTOs/Diskless-HOWTO-3.html. [Online]. Available: http://linuxdocs.org/HOWTOs/Diskless-HOWTO-3.html
[3] "Operating system market share," http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10. [Online]. Available: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10
[4] "Operating systems usage / market share (Trend analysis)," http://www.statowl.com/operating_system_market_share_trend.php. [Online]. Available: http://www.statowl.com/operating_system_market_share_trend.php
[5] O. H. Alliance. (2010) Home - android - overview. Accessed: October 1, 2012. [Online]. Available: http://www.openhandsetalliance.com/android_overview.html
[6] [Online]. Available: http://www.ltsp.org/
[7] "Zentyal – the linux small business server." [Online]. Available: http://www.zentyal.com/es/
[8] "LXDE.org | lightweight x11 desktop environment," http://lxde.org/, 2009. [Online]. Available: http://lxde.org/
[9] E. Y. Chen, "Virtual smartphone over ip," June 2010.
[10] M. T. Jones, "Linux initial RAM disk (initrd) overview," Jul. 2006. [Online]. Available: http://www.ibm.com/developerworks/linux/library/l-initrd/index.html
[11] P. T. Breuer, Arturo García Ares, and A. Martín Lopez, "The network block device | linux journal," http://www.linuxjournal.com/article/3778, May 2000. [Online]. Available: http://www.linuxjournal.com/article/3778
[12] A. x86 Project. (2012, 02) Android x86 - 4.0.3-RC1 - Release Notes. [Online]. Available: http://www.android-x86.org/releases/releasenote-4-0-rc1.
[13] T. Kay, "Linux swap space," http://www.linuxjournal.com/article/10678?page=0,1, Mar. 2011. [Online]. Available: http://www.linuxjournal.com/article/10678?page=0,1
[14] A. Developers, "What is android?" http://developer.android.com/guide/basics/what-is-android.html. [Online]. Available: http://developer.android.com/guide/basics/what-is-android.html
[15] "Android debug bridge | android developers." [Online]. Available: http://developer.android.com/guide/developing/tools/adb.html
[16] "Summary of currently included benchmark cases - comprehensive benchmark suite for android," http://code.google.com/p/0xbench/wiki/Benchmarks. [Online]. Available: http://code.google.com/p/0xbench/wiki/Benchmarks
[17] "Comparison of free and paid android apps - AppBrain." [Online]. Available: http://www.appbrain.com/stats/free-and-paid-android-applications
[18] "Number of available android applications - AppBrain." [Online]. Available: http://www.appbrain.com/stats/number-of-android-apps
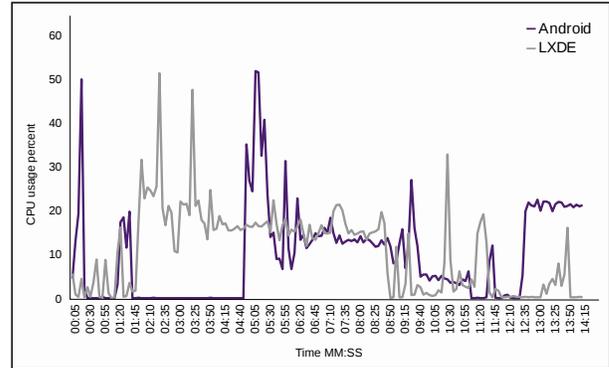
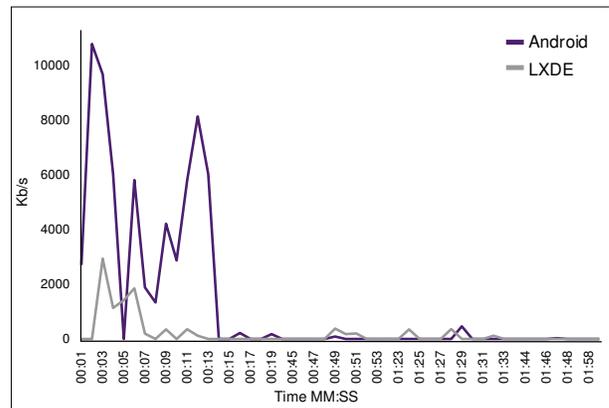Fig. 7: CPU statistics on diskless nodes experiment.
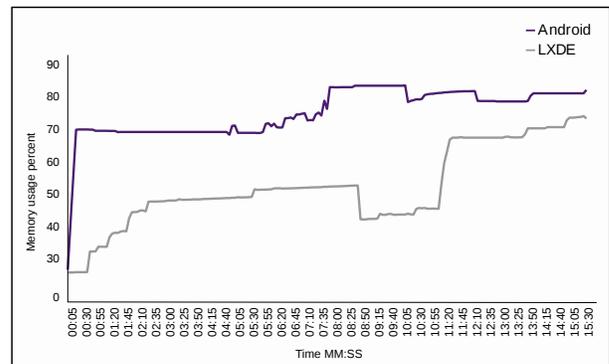

Fig. 8: I/O operations on diskless nodes experiment.

Fig. 9: Memory statistics on diskless nodes experiment.