Remote Debugging On Target Devices

Kasturi Rangan R
M. Tech, SJCE Mysuru

Abstract— *The main aim is to achieve remote debugging of multiple target devices connected to server by the multiple users. This solves the hardware shipment problem for debugging which is more economic. ADB is used to debug the android target device and GUI is created to make debug easy by the multiple users. This GUI will also act as the manager to control the multiple users over the devices connected by allocating, deallocating devices to the user.*

***Index Terms*—**VNC (virtual network computing); ADB; Remote Debugging; Target Devices.

I. INTRODUCTION

In the beginning, there was the debug monitor. Inexpensive but effective, it still serves handily alongside the most expensive debugging tools. Yet monitors have their drawbacks and weaknesses. For instance, they require ROM, RAM, and a communications channel from the target; they need to be ported to the target hardware; and they don't let you set breakpoints in programs running out of ROM.

After the debug monitor came the in-circuit emulator (ICE). By applying some clever hardware methods (usually based on special bond-out versions of processors), an ICE provides capabilities far beyond those of a simple ROM monitor. The ICE's most powerful features include complex breakpoints (even in ROM), real-time traces of processor activity, and no use of target resources. But this extra functionality comes at a high cost.

To counter these trends, many semiconductor vendors now integrate dedicated debug circuitry into their chips. Each vendor generally has its own proprietary name for this technology, such as BDM, OnCE, and MPSD. Other vendors simply add software debug capabilities to their existing JTAG ports. Collectively, we'll call these technologies on-chip debug.

Remote Debugging: This technique involves running two debuggers at different locations. The debugger that is actually doing the debugging is called the Debugging server. The debugger that is controlling the session from a distance is called the Debugging client.

Server side:-The target devices are connected or located and are connected to server for which clients get the access.

Client side:-The user will be present and will be accessing the target device.

Tunnel / network:-The way the connection is setup is through this tunnel and the session will be managed by this network.

In debugging we have two types they are software debugging process and hardware debugging process. The hardware debugging is more economic when compared to software debugging because of hardware shipment. In this context Hardware shipment is the process of sending the hardware devices from one location to another for debugging. For solving the hardware failures or to debug the hardware/target devices the hardware independent debugging process is needed which avoids hardware dependency.

II. LITERATURE SURVEY

A **debugger** or **debugging tool** [1] is a computer program that is used to test and debug other programs (the "target" program).

The code to be examined might alternatively be running on an Instruction set simulator (ISS), a technique that allows great power in its ability to halt when specific conditions are encountered but which will typically be somewhat slower than executing the code directly on the appropriate (or the same) processor. Some debuggers offer two modes of operation full or partial simulation to limit this impact.

A "trap" occurs when the program cannot normally continue because of a programming bug or invalid data. For example, the program might have tried to use an instruction not available on the current version of the CPU or attempted to access unavailable or protected memory. When the program "traps" or reaches a preset condition, the debugger typically shows the location in the original code if it is a **source-level debugger** or **symbolic debugger**, commonly now seen in integrated development environments. If it is a **low-level debugger** or a **machine-language debugger** it shows the line in the disassembly (unless it also has online access to the original source code and can display the appropriate section of code from the assembly or compilation).

Some of the most capable and popular debuggers implement only a simple command line interface (CLI)—often to maximize portability and minimize resource consumption. Developers typically consider debugging via a graphical user interface (GUI) easier and more productive. This is the reason for visual front-ends that allow users to monitor and control subservient CLI-only debuggers via graphical user interface. Some GUI debugger front-ends are designed to be compatible with a variety of CLI-only debuggers, while others are targeted at one specific debugger.

A. ADB (Android debugging bridge)

Android Debug Bridge (ADB) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:

- A client, which runs on your development machine. You can invoke a client from a shell by issuing an ADB command. Other Android tools such as the ADT plugin and DDMS also create ADB clients.

- A server, which runs as a background process on your development machine. The server manages communication between the client and the ADB daemon running on an emulator or device.

- A daemon, which runs as a background process on each emulator or device instance.

B. VNC

Virtual Network Computing (VNC) [2] is a graphical remote control desktop sharing software which allows you to view and fully interact with one computer desktop (running a "VNC server") using a simple program (the "VNC viewer"). The viewer is run from another computer desktop anywhere on the Internet. The two computers don't even have to be the same type, i.e. the system is independent of the platform and operating system, so for example you can use VNC to view a SUSE-11 desktop at the office from a Linux/windows/mac computer at home or at another location. Moreover, many viewers can simultaneously display that same desktop of the running server session. Keyboard and mouse events are transmitted between the client and host computers over a network[3].

In summary, a VNC system consists of a Server (Xvnc)[4], a Client (vncviewer), and a Communication Protocol:

- The VNC Server (Xvnc, invoke with: vncserver) is the running service that shares its virtual screen, it is controlled by a client.
- The VNC Client (vncviewer) is the program that watches, controls, and interacts with the server.
- The VNC protocol (RFB) is a simple "Thin-Client" protocol, based on one graphic primitive from server to client, where it "puts a rectangle of pixel data at the specified X,Y position", and events the messages from client to server.

III. PROPOSED SOLUTION

The remote debugging on target devices framework mainly consists of two major components, they are
A. VNC network creation
B. Access control manager

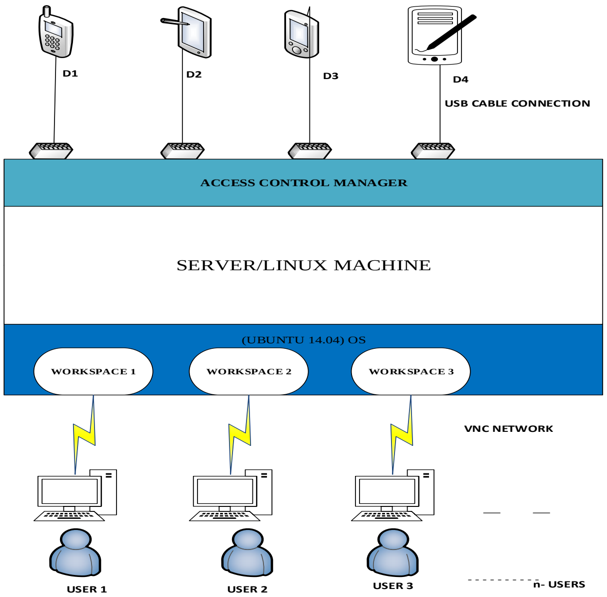

Fig. 1 Remote debugging on target devices framework which has major layers like VNC and Access control manager.

A. VNC network creation

It is used to connect and control one computer from other. It consists of two components they are VNC server and VNC client (viewer). Remote connection is achieved by using VNC. Before going to VNC server the workstations are to be created for different users so that users won't get conflict between them for resources. The steps in achieving the VNC network with independent users workstation is as follows.

a. VNC server installation
The dependencies which are required for the VNC server establishment are installed in the Unix server so that it helps in VNC network creation.

b. Add user in server
By using unix commands add user, give the details required to create user and later run vncserver and modify the xstartup file according to requirement

c. Start-up script for VNC server
This script is written and it should be run for calling the configuration file with the ports assigning to the users for their respective workstation. Without running this script VNC connection cannot be established.

d. Configuring VNCserver_conf file
This file is called by the vncserver start-up script for the information of the user and the desktop view configuration. It is must from which user with respect

to the workstation is identified and help in establishing Vnc n/w.

e. Install VNC viewer at client side
At the client side client should install the VNC viewer so that they can connect to the VNC server where their workstation has been created.

B. Access control manager

In implementation of Device handling (Access control manager) we have two ways

a) command prompt level (running scripts)
b) GUI (Graphical User Interface) with scripts and ADB commands.

a) Command prompt level:
The Perl scripts are used at the **Access control manager** layer in order to control the user access to the device and the execution of ADB commands. The Perl scripts [6] takes the input from the GUI and makes the decision for monitoring the devices.

There are 3 set of scripts
a. Server_side script
b. clientside_allocating script
c. clientside_deallocation script

The Sever_side script are mainly concentrating on the getting the info of the device and provide proper information to the user at the server end by interacting to GUI. This is required in order to reset device or handling device can be done at server end.

The clientside_allocating script involves in the allocation process of device to the users who are having the workspace in the main server.

The clientside_deallocating script involves in the deallocation process of device to the users who are having the devices and working in their workspace in the main server.

❖ ALGORITHM

- Server side script :

Steps:

1) Get the devices connected to the server and put into an array.
2) And by using embedded ADB command fetch the device name.
3) Later display device id and its name on standard o/p.
4) Push the device id with the flag initialize to zero to the file.

- Client side Script :

Steps:

1) Open the device list repository.
2) Indexing the devices and put up the device list with their index and flag into another repository.
3) Display all device id, index and flag respectively in the prompt.
4) Get the device index from user which he requires for debugging.
5) Allocate the device to user under his name.
6) Update the flag of the device in main repository.

- Client side close script :

Steps:

1) Get the device index of the device which the user had got access.
2) Verify with the user name taken during the allocation time.
3) Deallocate the device by changing the flag of device setting to zero.
4) Update the main file too.

b) GUI with scripts and ADB commands:

- Server side GUI :

Features:

➢ It shows the devices connected to server by its name and device ID.
➢ It shows the current status of the device flags.
➢ It refreshes the flag values to ‘0’ so that all devices are available and it also updates if the new device is added or any device removed.

- Client side GUI :

I. Welcome Window:

Features:
➢ Allocation button, which pop up the allocation window for device allocation.
➢ ADB windows which are the actual debugging windows with respect to the device allocated to the user.
➢ Deallocation button, which is used to deallocate the device.
➢ There is display of the information of the device allocated to user and not yet deallocated, which cautions the user not get allocate the new device before deallocating.

II. Allocation Window:

Features:

- ➢ It displays the list of devices which are available means not allocated to any user.
- ➢ By selecting the device ID and clicking the Allocate button makes the device available to the respective user.
- ➢ As soon as this allocate button is clicked the Perl script makes the flag of the device to high in main repository.

III. ADB Windows:

Features:

- ➢ Droid view which gives the allocated device display.
- ➢ Basic ADB commands are at single click with respect to allocated device.
- ➢ Argument ADB commands by providing argument get the activity done. Activities like Install, push, pull, play & removing files can be done.
- ➢ Mainly here we are able to get the bug report of the device and also the logcat of the allocated device.
- ➢ It provides the services list and the respective service info of the device allocated to the user.
- ➢ ADB shell argument commands which is much more helper to debug and to understand the properties and the values set to the device registers.

IV. Deallocation Window:

Features:

- ➢ Deallocates which means set the device flag to 0 so that it is available for other users.
- ➢ This will be updated at main repository too so it is like resource surrendering.

❖ ALGORITHM

Generic Steps Followed In GUI:

1) For each button there will be creation of process
2) This process has the Perl script running at back end.
3) Where these Perl scripts are embedded with the ADB commands.
4) ADB daemon will be running at the device end and ADB server will be running at server side.
5) The ADB commands to device gives the output, which will be displayed in the text box of the GUI.

IV. CONCLUSION

On the basis of the system test results, the tool can be used for the multiple users handling for the debug of android devices connected to the server machine. This process of remote debugging can be extended to other OS devices too and that is under future scope. So this process mainly reduces the cost of hardware shipment risks, either it may be lost of device or cost of shipping etc.

References

[1] Robert Love, “Debugging,” in Linux Kernel Development, 3rd ed.,
USA

[2] Otto Carlos M.B. Durate, Guy Pujolle, “Virtual Networks”, 1st ed, Brazil
[online]:
http://onlinelibrary.wiley.com/book/10.1002/9781118576946

[3] Handling Multiple VNCsessions
[online]:http://www.golinuxhub.com/2013/02/running-multiple-vnc-server-sessions.html

[4] XVNC learning
[online]http://www.hep.phy.cam.ac.uk/vnc_docs/xvnc.html(xvnc doc wiki)

[5] ADB learning [online] http://developer.android.com/tools/help/adb.html

[6] Perl scripting [online] http://www.perlmonks.org

[7] Ubuntu learning [online]:http://ubuntuforums.org