

Power Consumption Monitoring, Performance Testing & GUI Testing with Tools

Organized by



Supported by

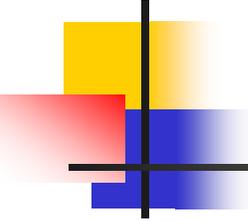


Funded by SME Development Fund



工業貿易署
Trade and Industry Department

Any opinions, findings, conclusions or recommendations expressed in this material/event (or by members of the Project team) do not reflect the views of the Government of the Hong Kong Special Administrative Region, Trade and Industry Department or the Vetting Committee for the SME Development Fund and the Dedicated Fund on Branding, Upgrading and Domestic Sales (Organisation Support Programme).



Topics



9:30AM-12:30PM

Performance Testing:
Introduction to Distributed
JMeter

Break

Performance Testing: Jmeter
Report and Jmeter Recording



2:00PM – 5:00PM

GUI Testing: Robotium and TestDroid

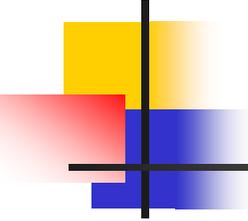
Break

Power Consumption: Battery
Historian

Hardware Requirements

- The participants should have
 - A real android devices
 - A USB cable used to connect to Windows 7





Lab Environment

- Windows 7 (64 bit , 16 GB RAM)
- Select Option [3]SME workshop in the boot menu when you start the computer
- Wifi:
 - Hotspot: SME_workshop or SME_workshop_5G
 - Password: \$mew0rk\$h0p
- All the files used in the workshop are located:
 - Local Disk D:/mobaptest_files/course_files/Session_1
 - Application APKs are in mobaptest_files/course_files/apk

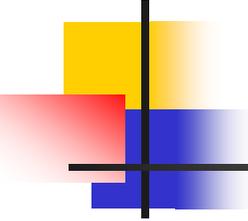
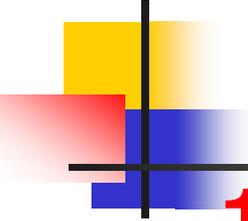


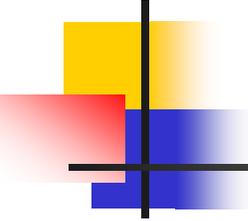
Table of Content

- Performance testing
 - Introduction to Distributed JMeter
 - Performance Testing: Jmeter Report and Recording Mobile Users action
- GUI testing
 - GUI Testing: Robotium Recorder and TestDroid Recorder
- Power consumption monitoring
 - Power Consumption: Battery Historian



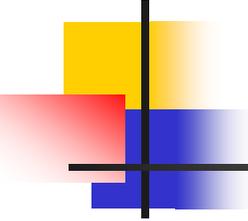
1 Introduction to Distributed JMeter





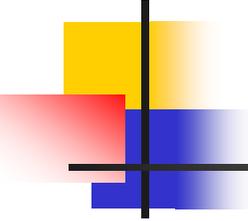
Brief History of JMeter

- An Apache project, open source software
- Java application developed for load testing
- Originally designed for testing Web application first released in 1998
- Has expanded to other test functions since released (e.g. webservices, ftp)



Platform

- Run on multiple platforms
 - Windows, Mac OS, Linux
- Can perform test on most of the platforms

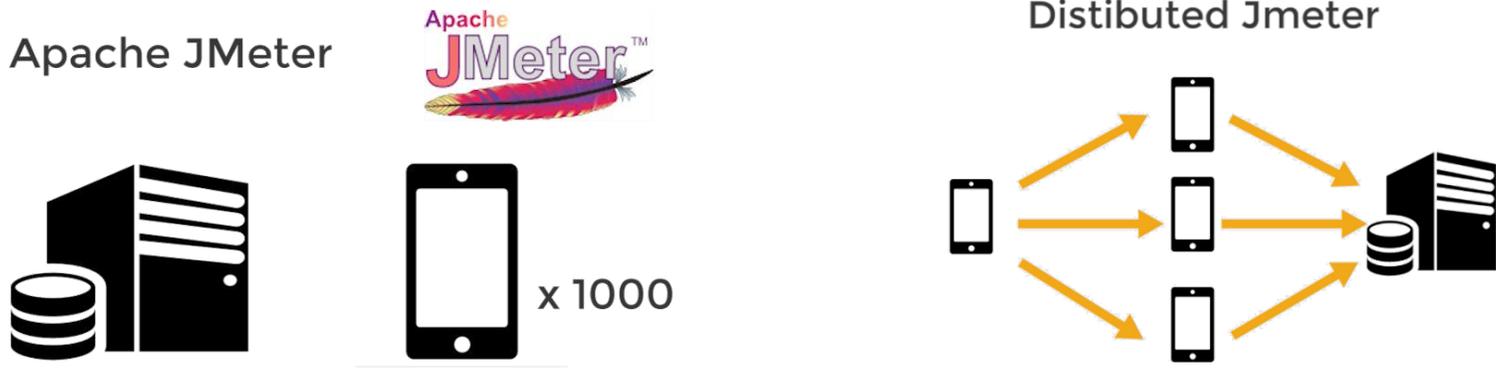


What to test

- Load Testing
- Simulate heavy loading on the server
- Capacity of the server
- JMeter can be used to test
 - Database connection,
 - FTP,
 - Webservices,
 - HTTP

Use Cases

- Testing the capacity of the mobile application server requires a huge amount of devices connected simultaneously!
- Automating the whole testing process without physical devices



Learning Objective #1

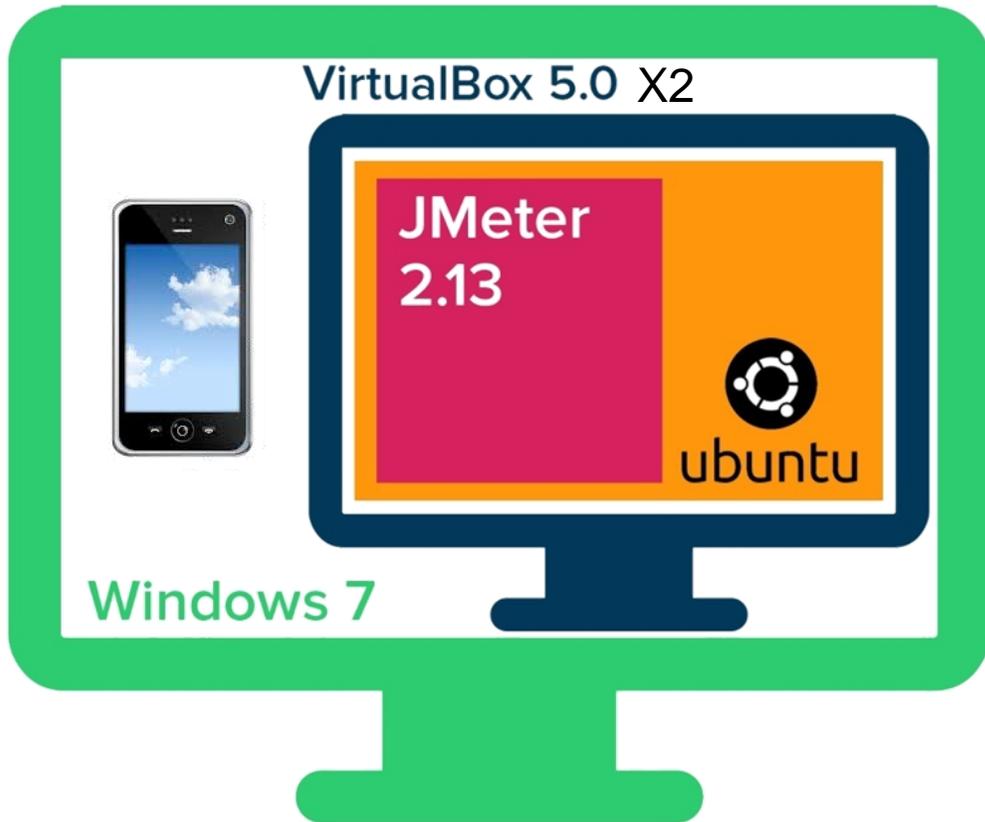
- Setup the testing environment of JMeter on one machine and perform a simple test case.

Apache JMeter

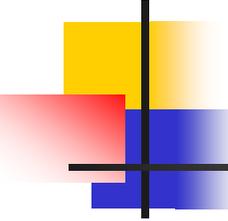


x 1000

Working Environment

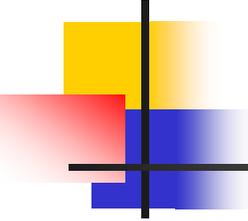


- We need to achieve that one physical machine with multiple virtual machines in order to perform a distributed testing solution
- A mobile emulator runs on Windows 7
- Virtual Box will be installed on Windows 7
- Two VMs will be setup . Ubuntu will operate in the VM and thus the JMeter can run on Ubuntu



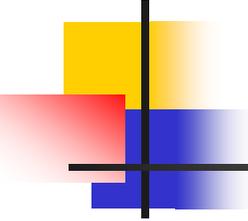
Installation: JMeter 2.13

- D:/mobaptest_files/course_files/Session_1/jmeter
- Prerequisite
 - VirtualBox 5.0.4 Installation
 - Ubuntu 14.04 Installation
 - Java 7 Environment setup on Ubuntu
- JMeter 2.13 Installation
- XAMPP (Apache+mySQL) 5.6.12 Installation
- Distributed Jmeter Setup
- *Note: pay attention on the version numbers*



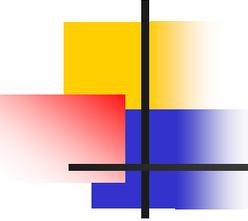
Installation: JMeter 2.13

- **preinstalled*
- VirtualBox 5.0.4 Installation
 - Download VirtualBox 5.0.4
 - <https://www.virtualbox.org>
 - Follow the instructions to install



Installation: JMeter 2.13

- Ubuntu 14.04 (64bit) Installation
 - Download Ubuntu 14.04 from
 - https://www.ubuntu.com/index_roadshow
 - Turn on VirtualBox and set the Ubuntu disk image to the CD drive of the VirtualBox
 - Follow the instruction to install Ubuntu
 - After installation, go to the VirtualBox menu and set the network adapter to **NAT Network**



Installation: JMeter 2.13

- Ubuntu 14.04 (64bit) Installation
 - In the VirtualBox tool bar
 - Select Devices > Install guest addition CD image
 - Then, in your virtual machine, install the loaded image
 - This will allow you to adjust the resolution of the virtual machine



Installation: JMeter 2.13

■ Java 7 Environment Setup

- Once the Ubuntu setup is finished, open terminal (Ctrl+Alt+T)
- Install OpenJDK Java 7 Runtime
 - Difference between OpenJDK and Oracle JDK:
 - <http://stackoverflow.com/questions/22358071/differences-between-oracle-jdk-and-open-jdk-and-garbage-collection>

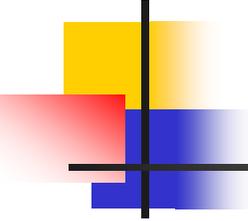
JDK 7

Debian, Ubuntu, etc.

On the command line, type:

```
$ sudo apt-get install openjdk-7-jre
```

The `openjdk-7-jre` package contains just the Java Runtime Environment. If you want to develop Java programs then install the `openjdk-7-jdk` package.

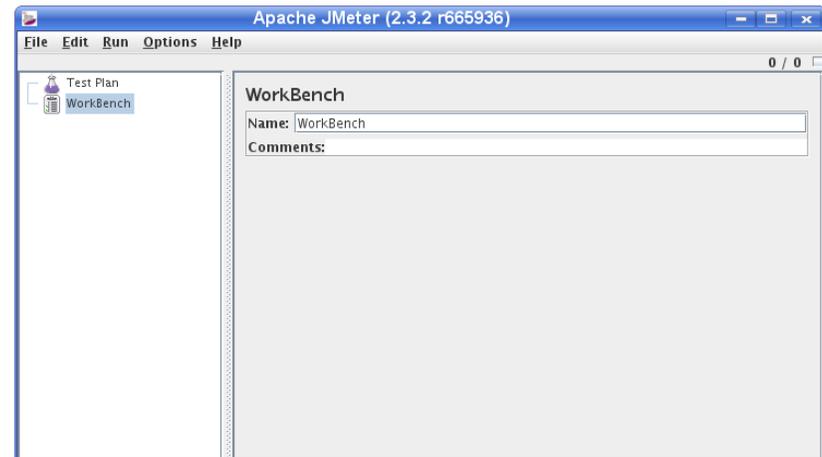


Installation: JMeter 2.13

- Download JMeter 2.13
 - http://jmeter.apache.org/download_jmeter.cgi
 - Or <http://www.mobaptest.org/app/apache-jmeter-2.13.zip>
- Decompress the file on the desktop

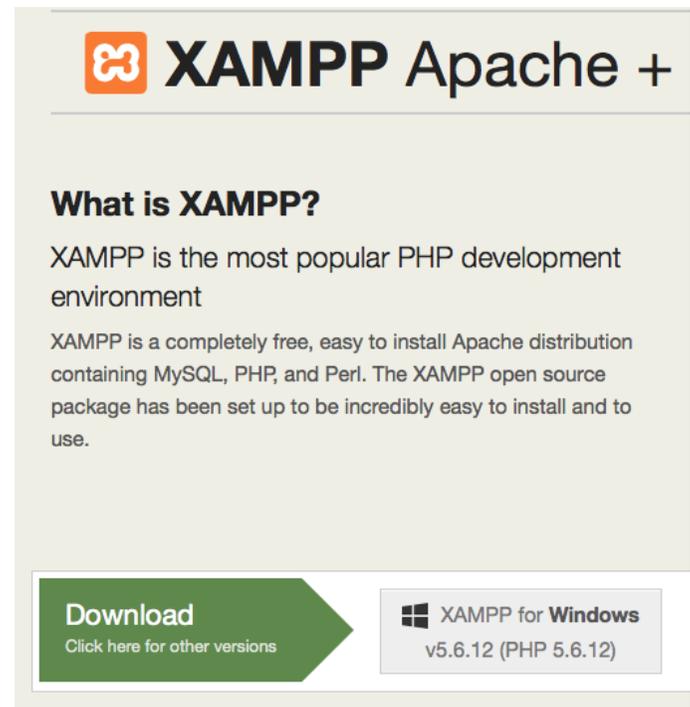
Jmeter Successfully Installed

- Open the terminal (Ctrl + Alt + T)
- Go to bin in the JMeter directory
 - `cd Desktop/apache-jmeter-2.13/bin`
- Run JMeter GUI
 - `java -jar ApacheJMeter.jar`



Installation: Apache Web Server

- **Pre-installed*
- Download XAMPP on Windows 7:
 - <https://www.apachefriends.org/index.html>
- XAMPP includes Apache web server, PHP development environment and MySQL...
- Leave setup option as default and install



XAMPP Apache +

What is XAMPP?

XAMPP is the most popular PHP development environment

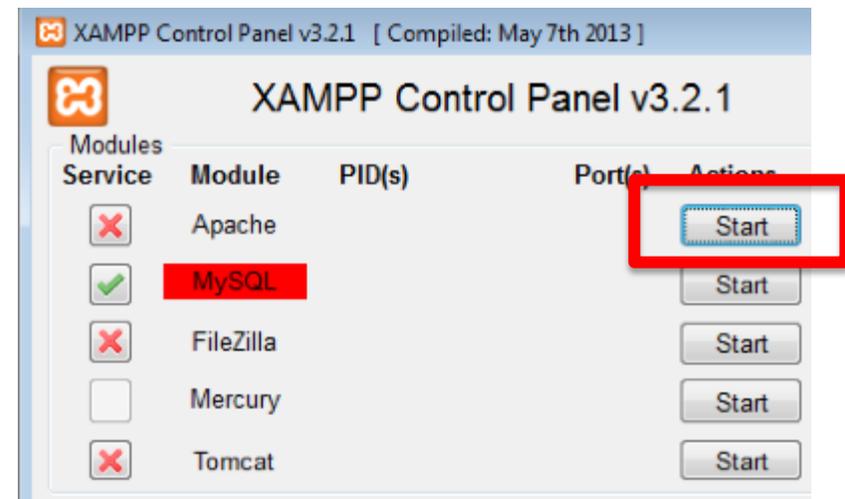
XAMPP is a completely free, easy to install Apache distribution containing MySQL, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

Download
Click here for other versions

XAMPP for Windows
v5.6.12 (PHP 5.6.12)

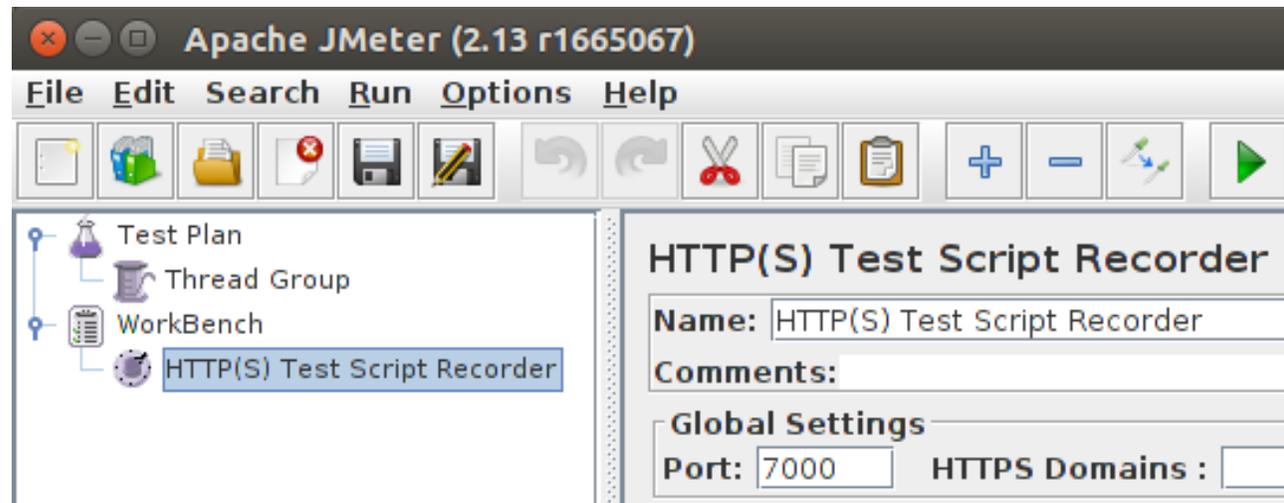
Installation: Apache Web Server

- After installation, open the XAMPP control panel
 - Right click on the shortcut icon and press "Run as administrator"
- Start the Apache Server



Setting proxy server: JMeter 2.13

- Right click on workbench:
 - Creating a proxy server: Test Script Recorder
 - Set the port number e.g. 7000



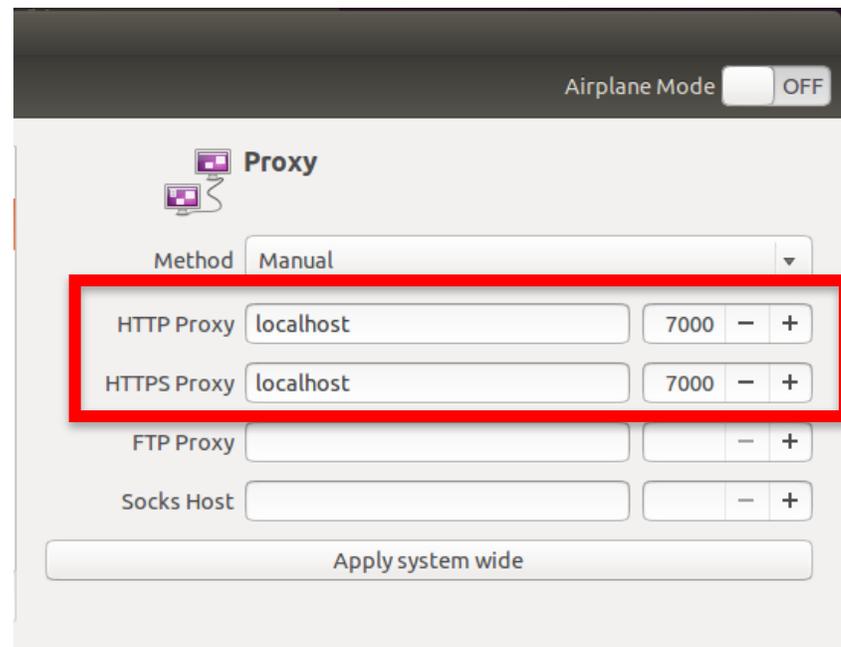
Setting proxy server: JMeter 2.13

- Configuring the proxy server:
 - In URL patterns to exclude section
 - Click the button, add suggested excludes
 - So the proxy server will filter the unwanted information
 - Start the proxy server



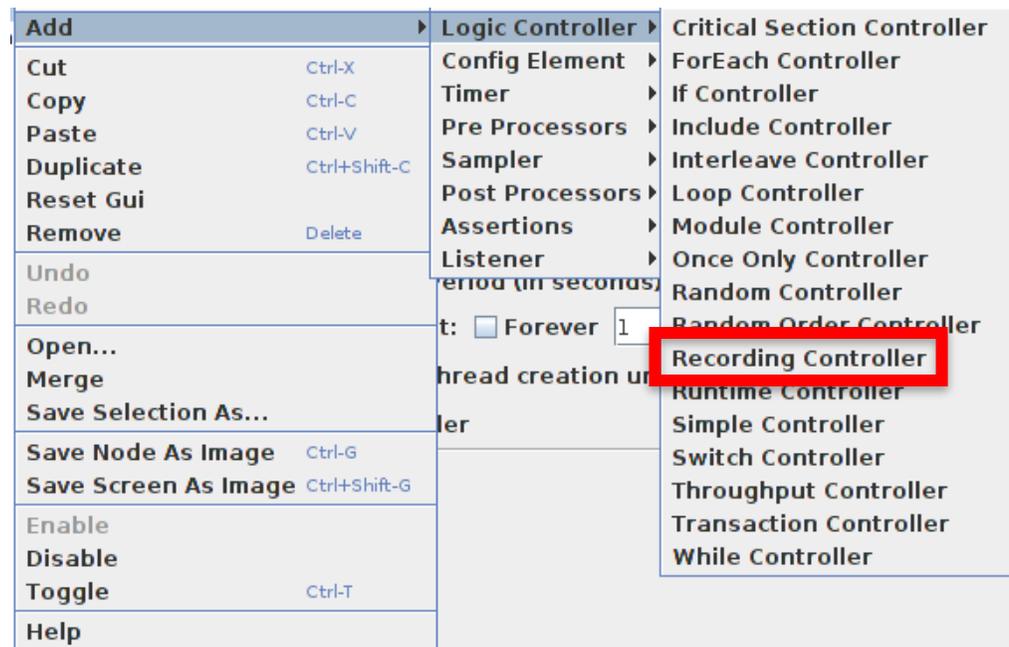
Setting proxy server: JMeter 2.13

- Set your browser to use the proxy server
 - Go to System Setting – Network – Network proxy
 - Set both the HTTP Proxy and HTTPS Proxy to: localhost: 7000



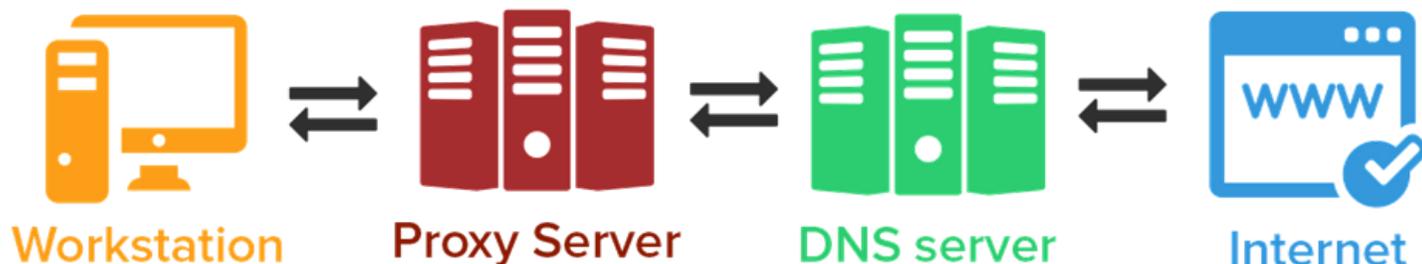
Recording HTTP actions - JMeter 2.13

- Right click on test plan:
 - Add a Thread Group
 - Add a Logic Controller, Recording Controller



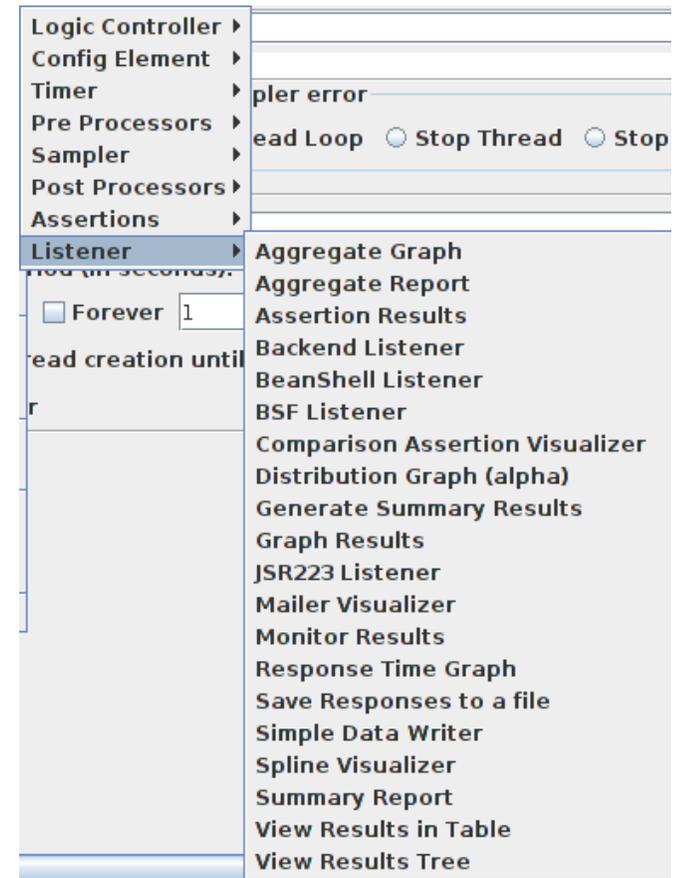
Recording HTTP actions - JMeter 2.13

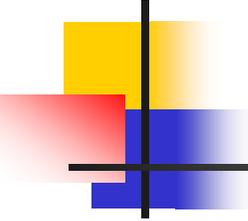
- Setting your browser to use the proxy server will allow the JMeter to view all the HTTP requests from the browser
- Recording controller will record all the HTTP request that go through the proxy server



Testing the Apache default page

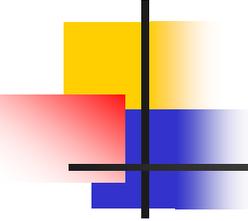
- Right click on Thread groups > Add > Listener > Summary Report
- Then you can view the result when you run the test





Testing the Apache default page

- Now you have setup the proxy server and the recording controller
- Try to access the Apache default page
 - Access the Windows 7 Apache server in your web browser – Enter the IP address on the browser
- Perform few actions on the page
 - E.g. Navigate different pages using the navigation bar
- Then stop the proxy server



Exercise #1: 5 mins

Our First Test Case

- Find out the IP of Windows 7
 - open command prompt
 - Enter: `ifconfig`
- Set up a simple test case on Jmeter to test the default page of Apache web server.
- Keep the default settings

Learning Objective #2

- Record the tester actions on a browser and allow Jmeter to perform the recorded actions thousand times at the same time

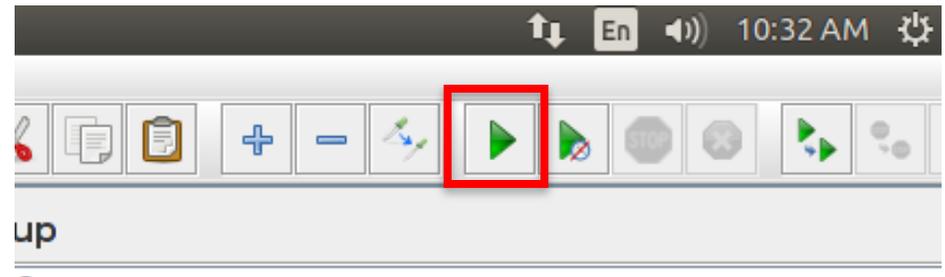
Apache JMeter

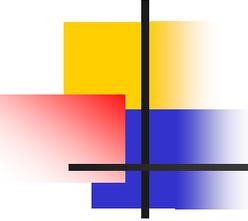


x 1000

Exercise #2 (5 minutes)

- Testing the Apache default page
 - After you perform all the actions, stop the proxy server
 - Set the number of threads in the thread groups tab to 50
 - Click the green arrow button on the tools bar to run the test



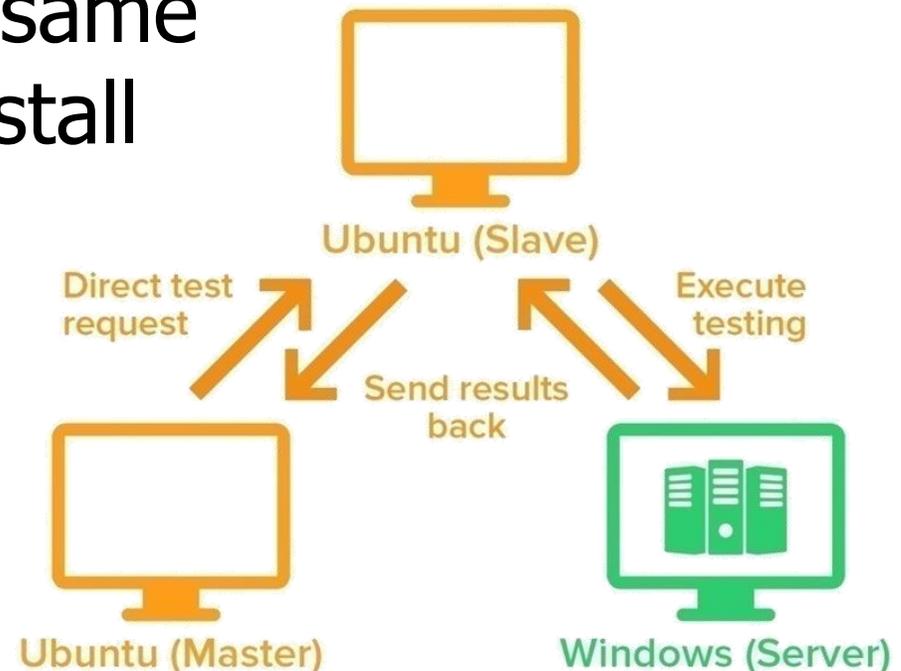


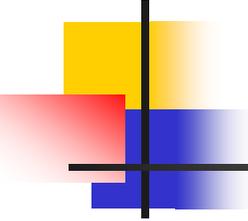
Setting up distributed JMeter - JMeter 2.13

- Two Virtual machines
- Two-step setup
 - 1. Start Slave System
 - 2. Start Master System

Setting up distributed JMeter - JMeter 2.13

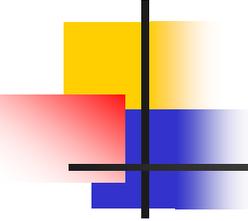
- Set up another VM and configure the Ubuntu with the same configuration, Install the Jmeter





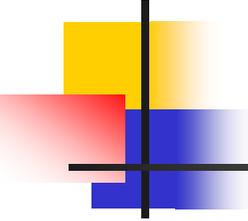
Setting up distributed JMeter - JMeter 2.13

- Start Slave System
 - Open terminal by (Ctrl + Alt + T)
 - Enter `$ifconfig`, mark down the ip address
 - E.g. 158.132.11.213
 - Change the accessing directory to
 - `$cd Desktop/apache-jmeter-2.13/bin`



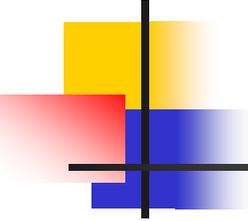
Setting up distributed JMeter - JMeter 2.13

- Start Slave System
 - Start the slave system by
 - `$./jmeter-server -Djava.rmi.server.hostname = 158.132.11.213`
 - Replace the ip address with the one you jotted
 - We should see:
 - Create remote object: UnicastServerRef[liveRef:.....]
 - Indicate the success of starting the remote system



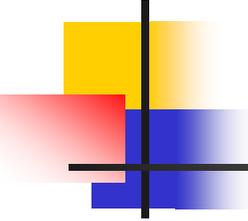
Setting up distributed JMeter - JMeter 2.13

- Start Master System
 - Go to your second virtual machine
 - Go to the Jmeter root directory. In apache-jmeter-2.13/bin, open jmeter.properties using the text editor
 - Look for the line:
 - Remote_hosts=127.0.0.1
 - Append the slave ip address to the end of the line
 - E.g. "remote_hosts = 127.0.0.1, 158.132.11.213"
 - Save the file



Starting distributed JMeter - JMeter 2.13

- Now your Slave system and the Master system should be setup correctly
- Open your Master system Jmeter GUI:
 - In terminal, enter:
 - `$cd Desktop/apache-jmeter-2.13/bin`
 - `$java -jar ApacheJMeter.jar`
- Run the distributed testing by:
 - Go to Menu – Run – Remote Start, click your slave system



Starting distributed JMeter - JMeter 2.13

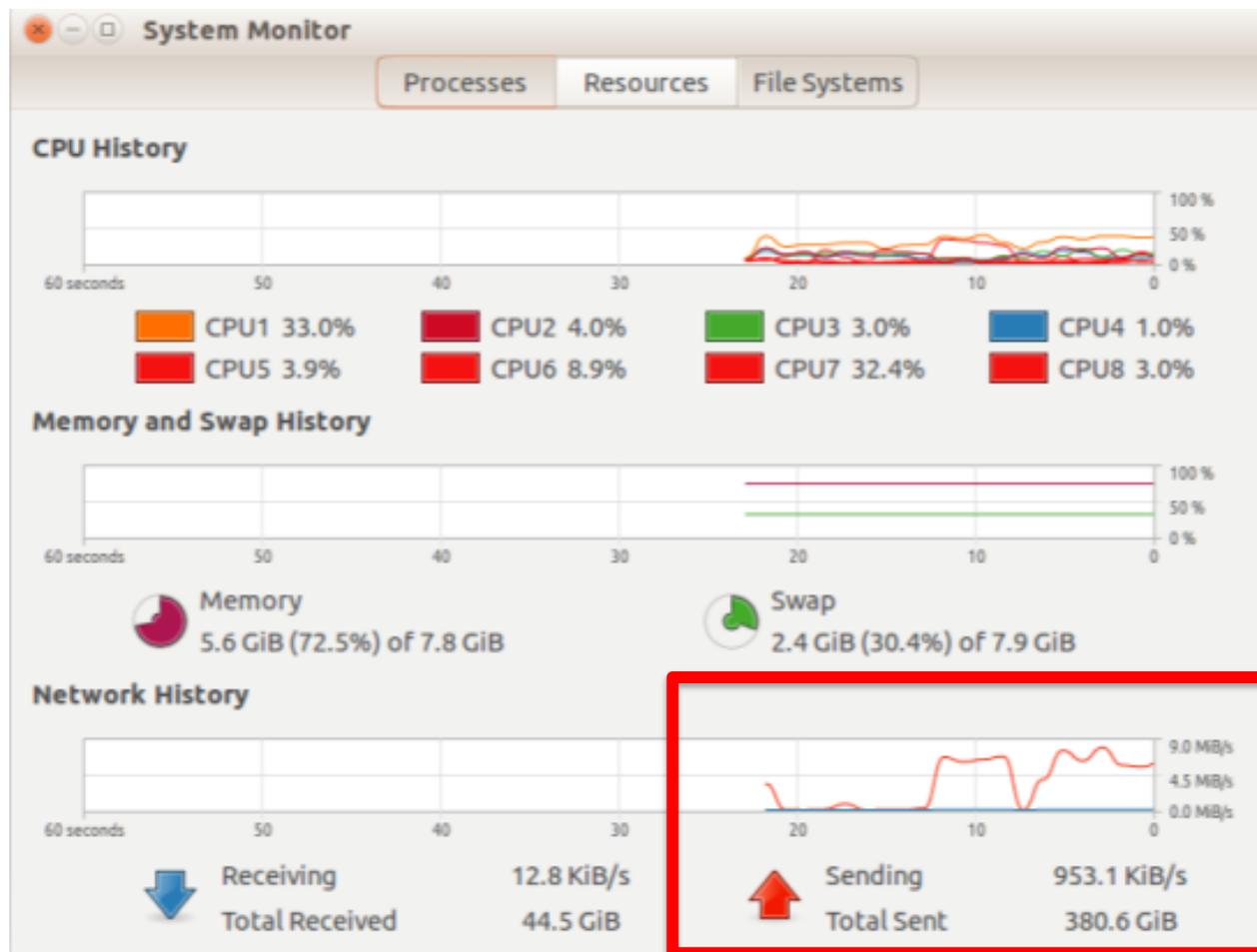
- Jmeter will ask you to save the test plan
- Click yes and save the file
- Since the test plan contains nothing, the test should finish in no time
- Suppose the remote testing is done successfully, you should see:
 - “Finished the test on host 158.132.11.213@....”
on your slave system terminal

Exercise #3 (5 mins) - JMeter

2.13

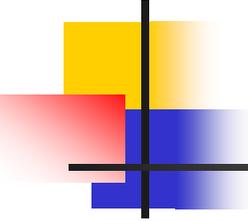
- Load your test plan recorded before (ref to #2)
- Run the remote testing for the default Apache page
- Increase the number of threads to 50 and loop forever
- Open the system monitor
- Should be able to see network traffic increases while testing

System monitor sample result



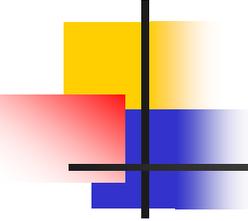
2 Performance Testing: Jmeter Report and Recording Mobile Users action





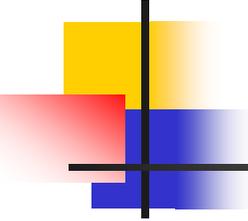
How to analyze the results - JMeter 2.13

- You can set the Jmeter to show different results by adding different listeners
- Jmeter can generate variety of reports to suit different needs
- We will introduce some of the essential graphs/ reports



Learning Objective #3

- Read the generated graphs
- Analyze the statistic recorded by the Jmeter to see on which area the web application / mobile application should improve



Report - JMeter 2.13 (Aggregate Report)

- Create a table that shows:
 - Response information of different requests
 - E.g. Max response time, Min response time, average response time
 - Responsive time = time requires for the server to response to the request sent
 - Throughput = Rate of successful message delivered over the network

Report - JMeter 2.13 (Aggregate Report)

- Gives a general reflection on how the server performs

Aggregate Report

Name:

Comments:

Write results to file / Read from file

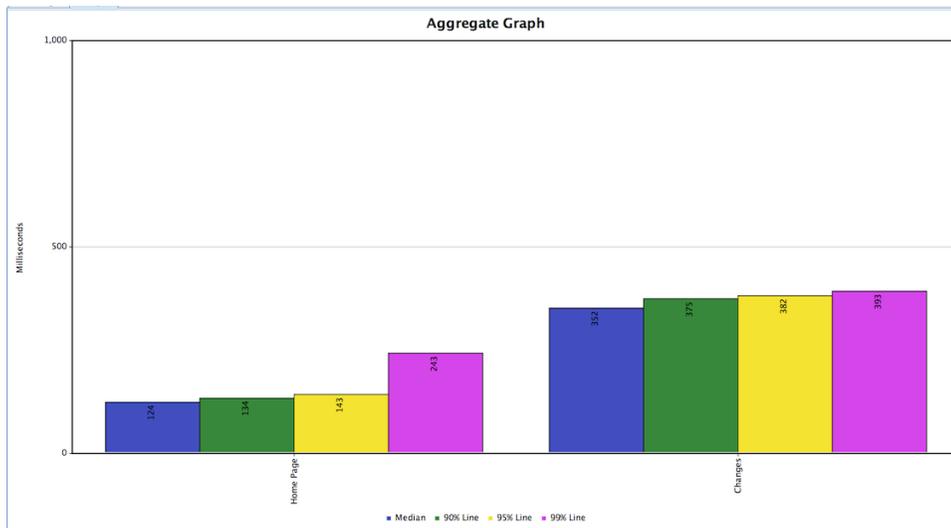
Filename Log/Display Only: Errors Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
Home Page	279	131	124	134	143	243	119	631	0.00%	8.9/sec	98.3
Changes	275	339	352	375	382	393	231	423	0.00%	8.8/sec	382.3
TOTAL	554	235	239	365	375	388	119	631	0.00%	17.5/sec	474.0

Include group name in label? Save Table Header

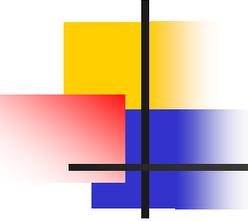
Report - JMeter 2.13 (Aggregate Graph)

- Provide a graph in addition to the aggregate report
- Easy to generate different bar graphs according to user's setting



Report - JMeter 2.13

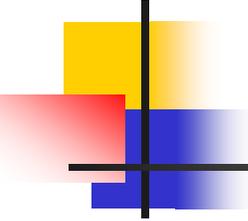
(Graph results)



- A simple graph that plots all sample times
- Provide overall graphs like median, average, deviation and throughput which integrated all the results
 - The abnormality from the population standard
- See next slide

Report - JMeter 2.13 (Graph results)



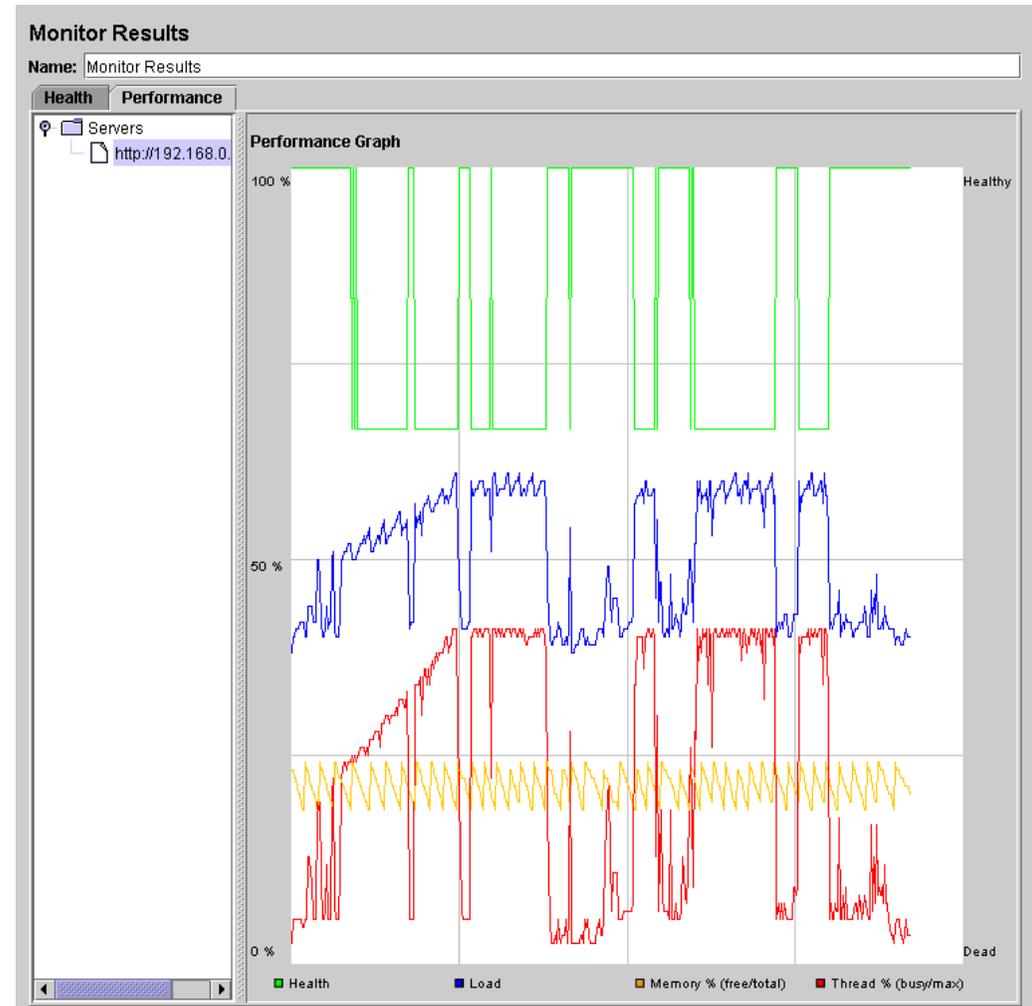


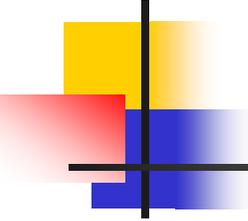
Report - JMeter 2.13 (Monitor Results)

- A graphic visualizer that display the server status
- 4 parameters of the server is plotted on the graph:
 - Health, Load, Memory, Thread
- Gives overall performance of the server

Report - JMeter 2.13 (Monitor Results)

- Green line represents the health of the server
- Blue represents the load on the server
- Red represents threads that accessing the server
- Yellow represents the memory usage of the server





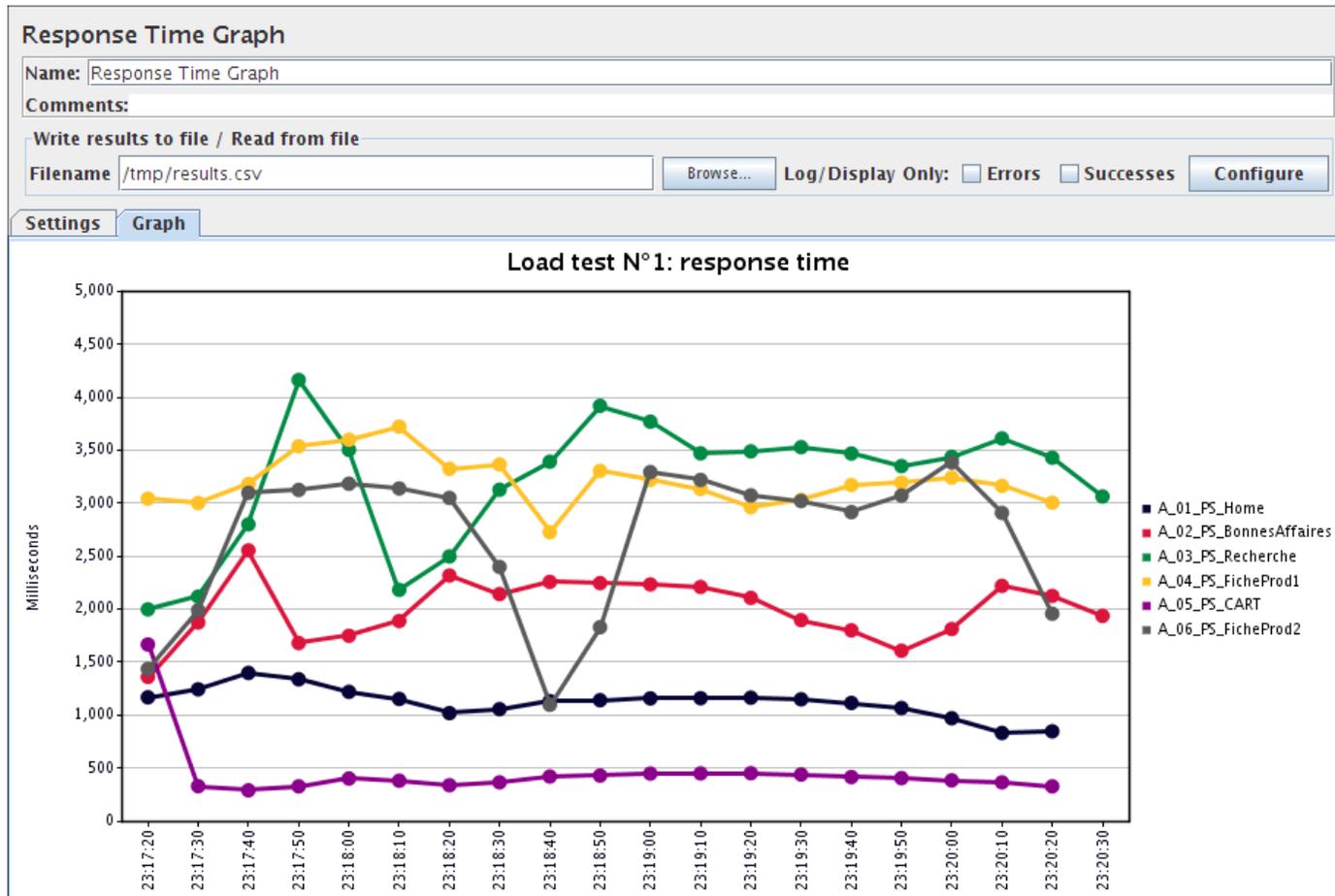
Report - JMeter 2.13

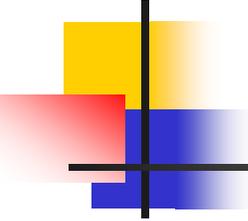
(Response time graph)

- Line chart showing the response time throughout the test
- Record every requests and plot on the graph
- If too many requests are recorded, a mean value will be plotted

Report - JMeter 2.13

(Response time graph)





Report - JMeter 2.13

(Summary report)

- Similar to Aggregate report, but use less memory
- Summary report listener does not keep a copy of every sample, while Aggregate report listener keeps a certain amount of copies for aggregation
- Summary report suits better if huge amount of samples are taken

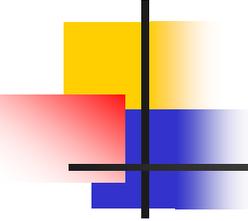
Learning Objective #4

- Record the tester actions on a mobile device and allow Jmeter to perform the recorded actions thousand times

Apache JMeter

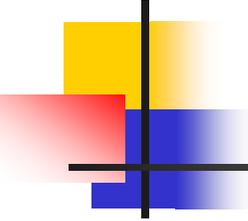


x 1000



Recording test plan - JMeter 2.13

- First copy the php folder locate in the jmeter course files to C:/xampp/htdocs
- So our app can reach the files in the local server



Recording test plan - JMeter 2.13

- Open JMeter GUI
- In test plan:
 - Add a Thread Group
 - Add a Logic Controller, Recording Controller
- In Workbench:
 - Creating a proxy server: Test Script Record
 - Set the port number and start the server

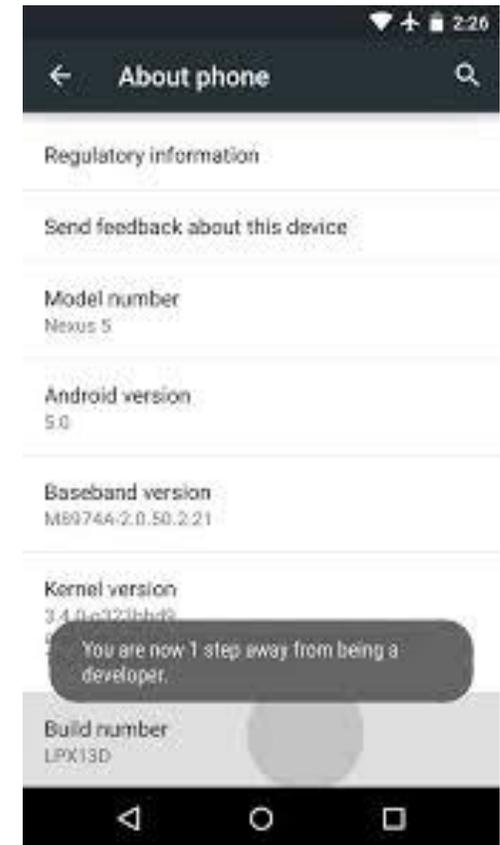
Recording test plan - JMeter 2.13

- Port forwarding: since we use NAT setting on our virtual machines, we need to forward the port for devices to reach the proxy server
- In the VM VirtualBox Manager, select File > Preferences > Network > Edit selected NAT network, then select port forwarding
- ****You need to delete the existing rule***
- Add a rule, set the host IP as 0.0.0.0 and host port 2222. The guest IP and guest port will be the VM IP and port



Recording test plan - JMeter 2.13

- You can use either use:
 - **Option 1:** an emulator to record the test plan
 - **Option 2:** a real device to record the test plan



Recording test plan - JMeter 2.13

- **Option 1:** Emulator setting:
 - Start your emulator using command prompt
 - Open command prompt and change directory to
 - `cd AppData/Local/Android/sdk/tools/`
 - Start the emulator by enter:
 - `emulator -avd (device_name) -no-audio -http-proxy (Host IP):(Host Port)`
 - Host IP should start with `127.x.x.x`

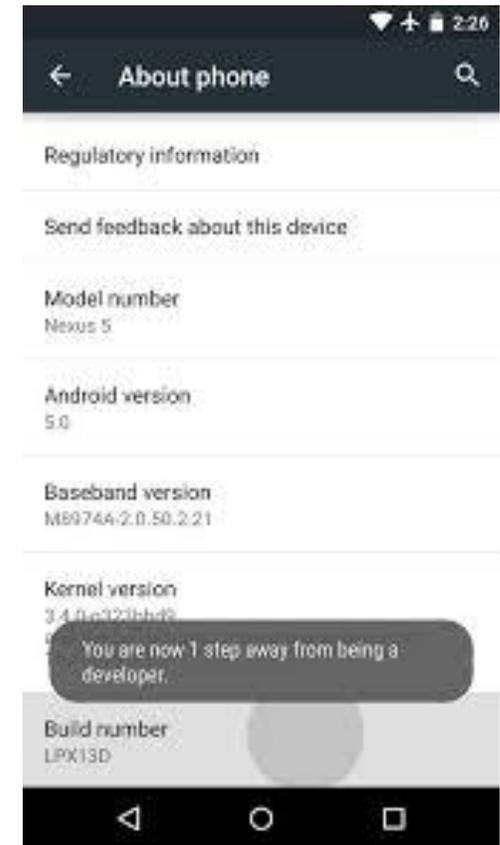
```
C:\Users\HKPUADMIN\AppData\Local\Android\sdk\tools>emulator -avd gary -no-audio
-http-proxy 158.132.11.96:7000

C:\Users\HKPUADMIN\AppData\Local\Android\sdk\tools>emulator: device fd:672

HAX is working and emulator runs in fast virt mode
creating window 0 0 480 800
```

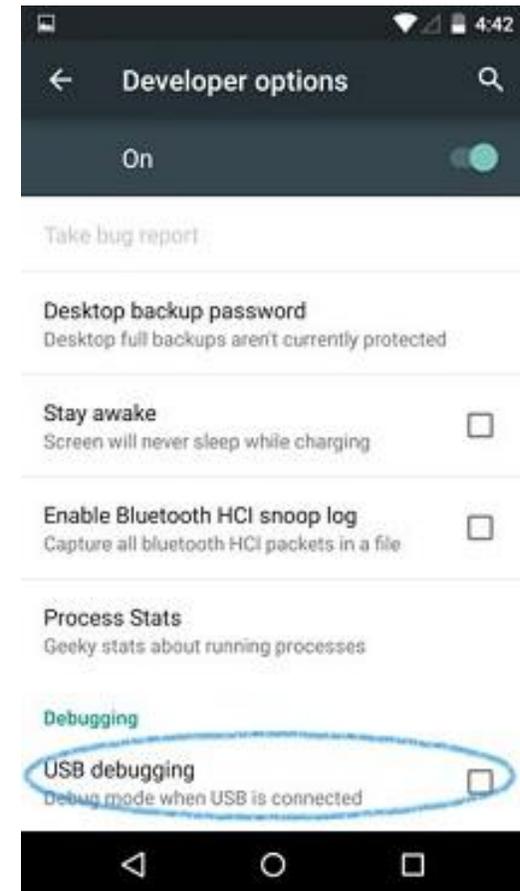
Recording test plan - JMeter 2.13

- **Option 2:** Real device setting:
 - Connect your device to the computer using WIFI
 - Enable developer setting in your device
 - Go to Settings > About phone
 - Scroll to bottom and find Build Number
 - Tap it seven times



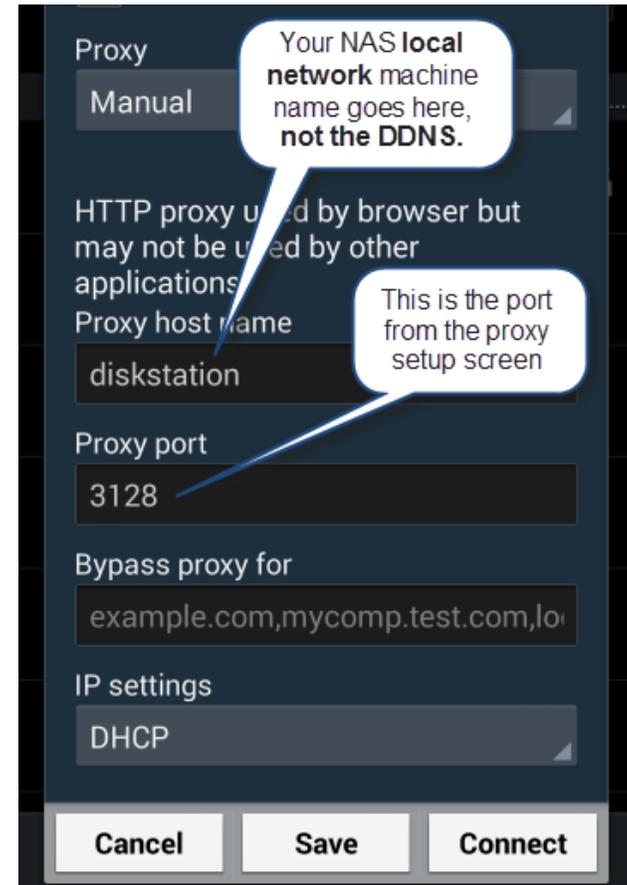
Recording test plan - JMeter 2.13

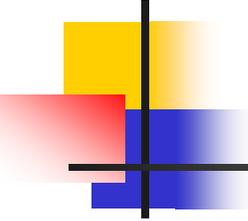
- **Option 2:** Real device setting:
 - Check the USB debugging dialog box in Developer options



Recording test plan - JMeter 2.13

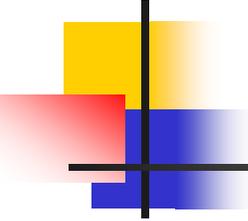
- **Option 2:** Real device setting:
 - Setup the device to use the proxy server created
 - Go to setting > WIFI
 - Long tap on the connected network's name
 - Modify network config > Show advanced options
 - Set proxy settings





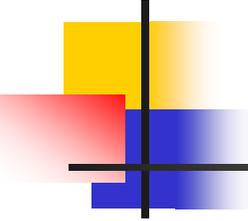
Recording Test Plans with mobile device- JMeter 2.13

- **Option 1:** Installing apk on emulator:
 - Turn on your emulator using Android Studio
 - Access the root directory of Android SDK
 - %USERPROFILE%/AppData/Local/Android/sdk/platform-tools
 - Right click while holding Shift > Open command window here
 - Enter: adb install example.apk
 - ADB (Android Debug Bridge) will install the indicated APK on your only running emulator



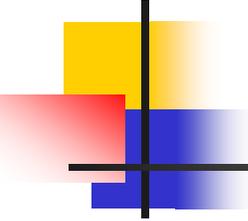
Recording Test Plans with mobile device- JMeter 2.13

- **Option 2:** Installing apk on real device:
 - Using PDANet, copy the apk to your Android device
 - Locate the apk using file manager on the device
 - Install the apk
- You may need to change the setting in:
 - Settings > Lock screen and security > Unknown sources
 - Enable installation of apps from sources other than the Play Store



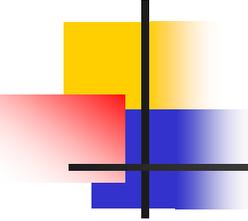
Recording test plan - JMeter 2.13

- Once you set up your proxy server on JMeter, every HTTP request will be recorded
- Perform the actions that need to be tested
- When you finished, stop the proxy server
- The test plan is recorded the recording controller
- Save the test plan



Executing test plan - JMeter 2.13

- Press on the Thread Group, set the number of threads (users)
- Add an Aggregate Report under Listener to the Thread Group (For viewing the results)
- Set the Loop Count to forever
- Run the test for 20 seconds then stop it



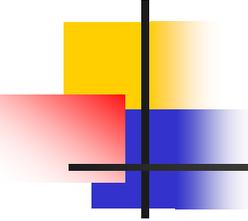
Recording Test Plans with mobile device- JMeter 2.13

- Connect your mobile device to the proxy server you set up
- Open your application to perform the actions that need to be tested
- Stop the proxy server once you finished

Exercise #4 (20 mins) - JMeter

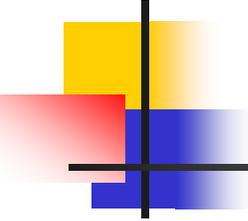
2.13

- Given the environment to run our E-stock apk, we would like to find out the limit of our server.
 - Install E-stock apk (can be found on the desktop folder) on the real device
 - Record the test plan with the application
 - Run the remote testing
 - Try to find the limit of the server by trying access the app with your emulator/real device while testing
 - Response become very slow = reached the limit
 - Answer



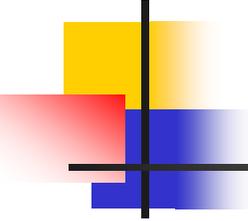
3 GUI Testing: Robotium Recorder and TestDroid Recorder





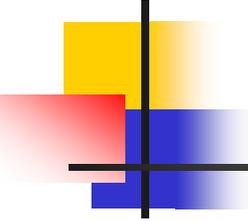
Brief History of TestDroid Recorder & Robotium Recorder

- Robotium is an open-source testing framework for Android applications
- Can be used both for testing:
 - Application with source code available or;
 - Applications where only APK file is available
- Both can be used to record test scripts for Robotium framework



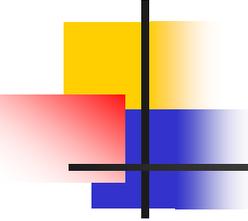
Platform

- TestDroid Recorder:
 - Windows/ MacOS/ Linux
 - Eclipse
- Robotium Recorder:
 - Windows/ MacOS/ Linux
 - Eclipse / Android Studio



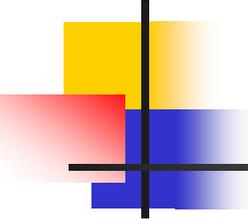
What to test

- Compatibility of the application on different devices with different size
- GUI stability of the application
- CPU usage of the application (TestDroid Cloud)



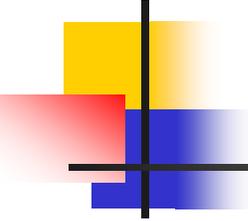
Use Cases

- Writing GUI testing script often needs to know the application's widget ID
- Can we record the user actions and generate the script which can be modified later
- Manual coding is inefficient, can TestDroid Recorder and Robotium Recorder help to solve this problem?



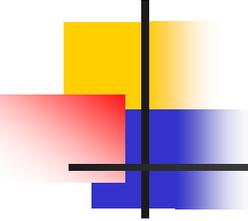
Learning Objective #5

- Install TestDroid Recorder and Robotium recorder
- Record a test script using each of the recorders



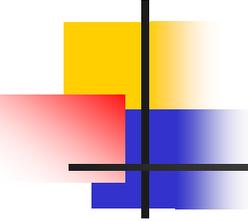
Installation: TestDroid Recorder

- D:/mobaptest_files/course_files/Session_1/testdroid_robotium
- Prerequisite:
 - TestDroid account
 - Java 8 Installation
 - Android SDK 24.3.4 installation
 - Eclipse Mars 4.5 Installation
 - ADT Plug-in installation
- TestDroid Recorder 4.5.3 Installaion



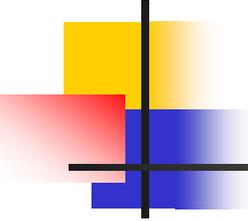
Installation: TestDroid Recorder

- Create your TestDroid account
 - Create the TestDroid account at:
 - <http://testdroid.com/try-for-free>
 - Follow the instructions to create the account



Installation: TestDroid Recorder

- Installing Java Environment
 - Download JDK 8 at Oracle
 - <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
 - Follow the instructions to install JDK 8

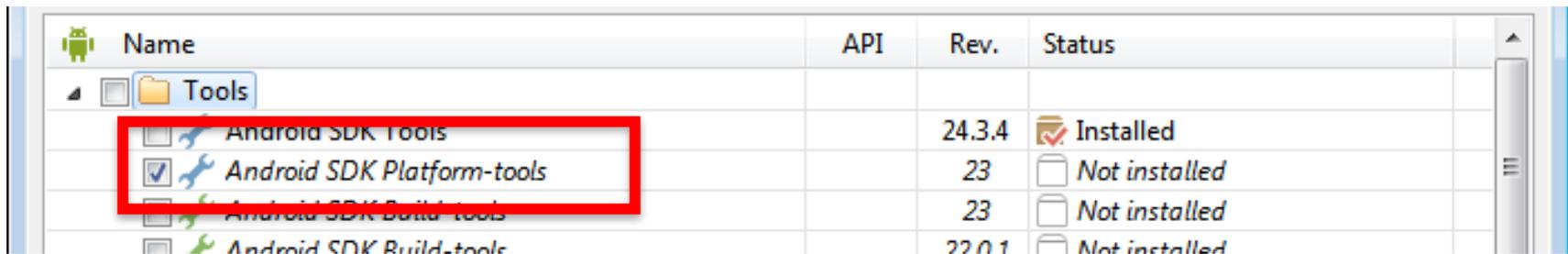


Installation: TestDroid Recorder

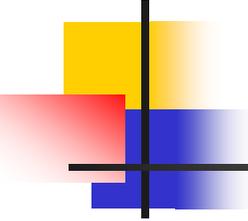
- Installing Android Debug Bridge
 - Android Debug Bridge comes with Android SDK
 - Download the stand-alone Android SDK 24.3.4
 - <http://developer.android.com/sdk/installing/index.html>
 - Start the SDK manager after installation

Installation: TestDroid Recorder

- Installing Android Debug Bridge
 - Uncheck all items
 - Check only the Android SDK platform-tools
 - Click Install package

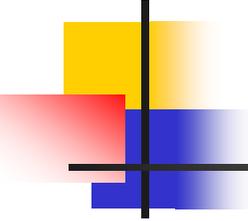


 Name	API	Rev.	Status
 Tools			
<input type="checkbox"/>  Android SDK Tools		24.3.4	<input checked="" type="checkbox"/> Installed
<input checked="" type="checkbox"/>  Android SDK Platform-tools		23	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android SDK Build-tools		23	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android SDK Build-tools		22.0.1	<input type="checkbox"/> Not installed



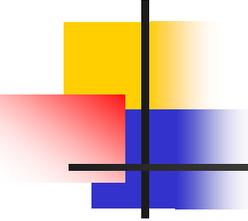
Installation: TestDroid Recorder

- Installing Eclipse Mars 4.5
 - Download Eclipse Mars 4.5
 - www.eclipse.org/downloads
 - Follow the instructions to install Eclipse Mars
 - After installing, start Eclipse
 - If Eclipse alert that SDK is not installed
 - Browse and specify the location of SDK
 - `C:/Users/virtualmachine/AppData/Local/Android/sdk`



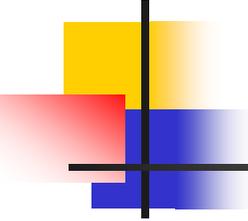
Installation: TestDroid Recorder

- Installing Android Development Tools (ADT) Plugin
 - Start Eclipse, then select Help > Install New Software
 - Click Add, in the top-right corner
 - Enter ADT Plugin as the name, the following as location:
 - <https://dl-ssl.google.com/android/eclipse/>
 - Click OK



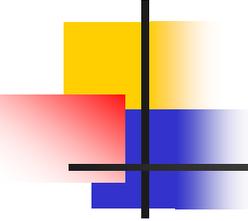
Installation: TestDroid Recorder

- Installing Android Development Tools (ADT) Plugin
 - Select the checkbox then click Next
 - Then you can see the list of tools to be downloaded, click Next
 - Click Finish



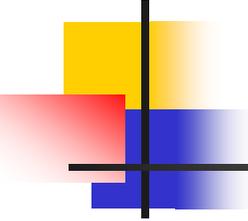
Installation: TestDroid Recorder

- Now you are set to install TestDroid Recorder
- Restart Eclipse, select Help > Install New software
- Click Add, enter
 - "TestDroid Recorder" as Name
 - "<http://www.testdroid.com/updates>" as location
- Check the checkbox and install the recorder



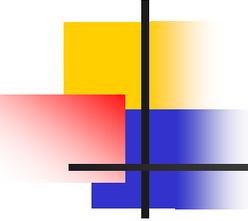
Starting: TestDroid Recorder

- Start Eclipse, select New > Other
- Select Android – TestDroid > First Recording
- Select My APK then browse to the apk location, click record
- Select your device then proceed to record
- Pause the record when you finished



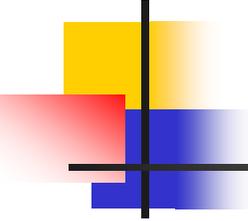
Exercise #5 (5 mins)

- Use the apk (see #3)
- Installed the provided SDK
- Record a test using TestDroid Recorder



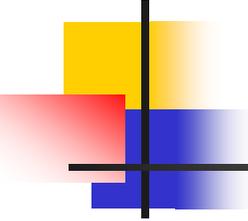
Installation: Robotium Recorder

- Installation of Robotium Recorder is very similar to that of TestDroid Recorder
- Prerequisite:
 - Java 8 Installation
 - Android SDK 24.3.4 installation
 - Eclipse Mars 4.5 Installation
 - ADT Plug-in installation
- Robotium Recorder Installation



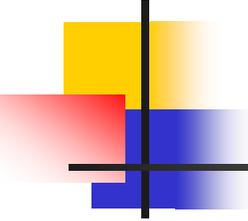
Installation: Robotium Recorder

- Prerequisite installations are the same
- To install Robotium Recorder in Eclipse:
 - Start Eclipse, select Help>Install new software
 - Enter the following in the “Work with” field
 - <http://recorder.robotium.com/updates>
 - Check Robotium Recorder and Uncheck Contact all update sites during install
 - Continue the installation
- Specify the JDK location when asked:
 - C:\Program Files\Java\jdk1.8.0_05



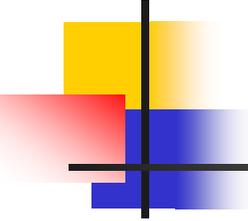
Starting: Robotium Recorder

- Start Eclipse, select New > Other
- Select Android – Robotium > New Robotium Test
- Select My APK then browse to the apk location, click record
- Select your device then proceed to record
- Pause the record when you finished



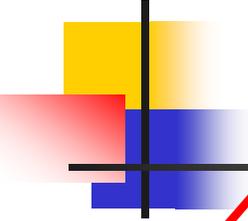
Exercise #6 (5 mins)

- Installed the provided SDK
- Record a test using Robotium Recorder



Exercise #7 (20 mins)

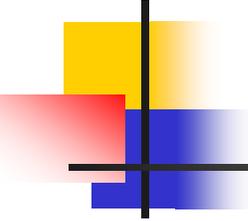
- Record the TestDroid script, modify the recorded script and execute the modified script
 - Installed the EStock APK
 - Record a test script testing the stock buying function
 - Run the test script and view the result
 - Then modify the script to run the test again



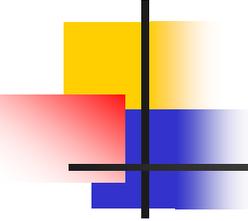
4 Power Consumption: Battery Historian



A Brief History of Battery Historian

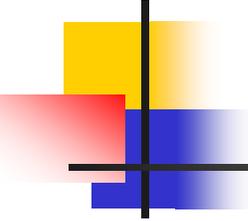


- Introduced along with Project Volta in 2014
- A part of Android Lollipop (5.0) aims at optimizing battery life
- Project Volta is to help developers improve battery performance



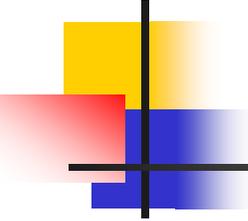
Platform

- Installation Platform: Windows 7
- Mobile Device: Android



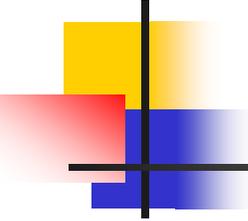
What to test

- Battery Historian 2.0 provides a battery usage visual timeline
- Developers can visualize battery usage at certain moment
- Pinpoint which function/hardware drains the battery when the app is operating



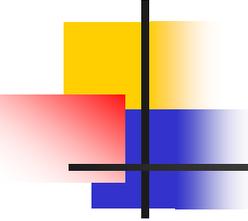
Use Cases

- Application users are alerted to the power consumption
- Application that uses power more efficient gives a better experience
- How to ensure that our app meets the user's expectation?



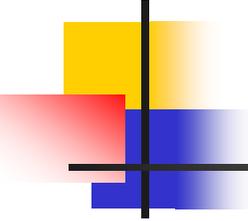
Learning Objective #6

- Install Battery Historian 2.0
- Generate bug report using Android device
- Import the bug report to view power consumption statistics



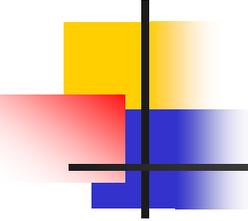
Installation: Battery Historian 2.0

- The installation involves number of steps
- Follow the instructions carefully
- `D:/mobaptest_files/course_files/Session_1/battery-historian`



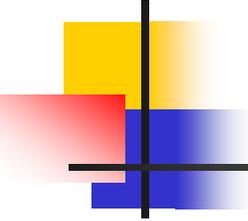
Installation: Battery Historian 2.0

- Prerequisite
 - Java 8 Environment
 - Android SDK 24.3.4 (adb)
 - Installing Git 1.9.5
 - Installing Python 2.7
 - Installing Go 1.4.2
- Installing Battery Historian 2.0



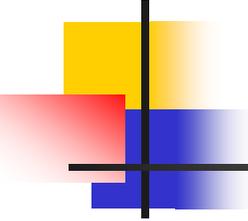
Installation: Battery Historian 2.0

- Installing Git 1.9.5
 - Downloading Git 1.9.5 at git-scm.com/download/win
 - Run the downloaded .exe



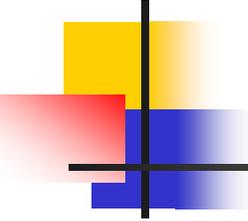
Installation : Battery Historian 2.0

- Installing Git 1.9.5
 - Click “Use Git from the Windows Command Prompt” option at the “Adjusting your PATH environment” step
 - Leave other option as default



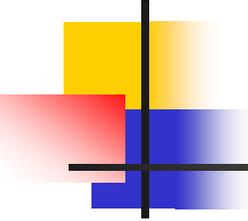
Installation: Battery Historian 2.0

- Installing Python 2.7
 - Download Python 2.7 from python.org
 - Choose the Windows x86-64 MSI installer to download
 - Follow the instructions to install Python 2.7



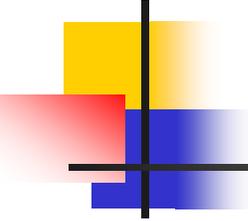
Installation: Battery Historian 2.0

- Installing Golang 1.4.2
 - Installing Golang programming language at golang.org/doc/install
 - Download go1.4.2.windows-amd64.msi for 64-bit Windows machine
 - Follow the instructions to install Golang 1.4.2



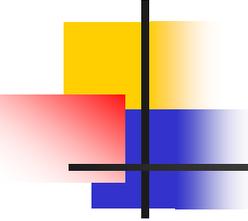
Installation: Battery Historian 2.0

- Installing Golang 1.4.2
 - Once the installation is done, set up a go workspace directory in your user profile folder (e.g. C:\admin\)
 - Set the path variable using the command prompt
 - set GOPATH=%USERPROFILE%\go-workspace
 - set GOBIN=%GOPATH%\bin
 - set PATH=%PATH%;%GOBIN%
 - cd %GOPATH%



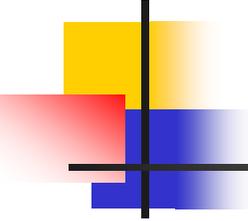
Installation: Battery Historian 2.0

- Installing Battery Historian 2.0
 - Install the battery-historian by
 - `go get -u github.com/golang/protobuf/proto`
 - `go get -u github.com/golang/protobuf/protoc-gen-go`
 - `go get -u github.com/google/battery-historian`



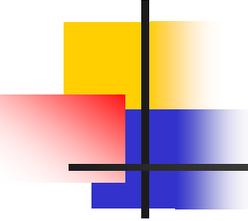
Installation: Battery Historian 2.0

- Installing Battery Historian 2.0
 - Then change directory to battery-historian folder
 - `cd src/github.com/google/battery-historian`
 - Go into battery-historian root:
`%GOPATH%/src/github.com/google/battery-historian`
 - create two folders: `third_party`, `compiled`



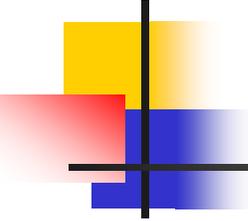
Installation: Battery Historian 2.0

- Installing Battery Historian 2.0
 - **Both folder available in course_files folder*
 - Download the Closure compiler and unzip it inside third-party/closure-compiler
 - <http://dl.google.com/closure-compiler/compiler-latest.zip>
 - Clone the closure library to third_party/closure-library using command prompt
 - `git clone https://github.com/google/closure-library third_party/closure-library`



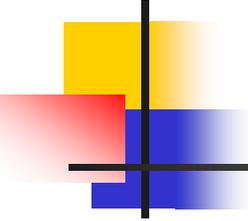
Installation: Battery Historian 2.0

- Installing Battery Historian 2.0
 - Finish the installation by these two commands
 - `third_party\closure-library\closure\bin\build\depswriter.py --root="third_party\closure-library \closure\goog" --root_with_prefix="js ../../../../js" > compiled\historian_deps-runfiles.js`
 - `java -jar third_party\closure-compiler\compiler.jar --closure_entry_point historian.Historian --js js*.js --js third_party\closure-library\closure\goog\base.js --js third_party\closure-library\closure\goog** --only_closure_dependencies --generate_exports --js_output_file compiled\historian-optimized.js --compilation_level SIMPLE_OPTIMIZATIONS`



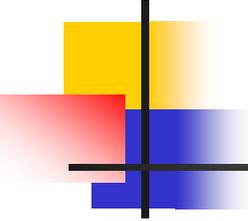
Installation: Battery Historian 2.0

- Installing Battery Historian 2.0
 - Start Battery Historian 2.0
 - `set GOPATH=%USERPROFILE%\go-workspace`
 - `cd %GOPATH%\src\github.com\google\battery-historian`
 - `go run cmd\battery-historian\battery-historian.go`
 - Open your web browser type:
 - `localhost:9999`
 - Default port is 9999



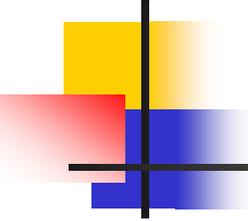
Running Battery Historian 2.0

- Set your phone on debugging mode
 - To enable usb debugging, go to Settings > About>Software information>More
 - Tap the Build number for 7 times
 - Go to Settings > Developer options, check the USB debugging box
- Connect your phone to the computer
 - If Windows does not recognize your phone:
 - Download PdaNet+ on Windows
 - Download PdaNet+ on Android
 - `course_files/Session_1/battery-historian/PdaNetA4191.exe`



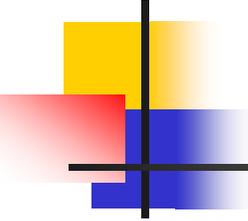
Running Battery Historian 2.0

- Go to the android platform-tools directory
 - `C:\%USERPROFILE%\AppData\Local\Android\android-sdk\platform-tools`
- Open command prompt in the directory



Running Battery Historian 2.0

- Check your phone connection by entering:
 - `adb devices`
- Reset Battery statistic data:
 - `adb shell dumpsys batterystats --reset`
- Unplug your phone and test the application
- Reconnect your phone back after testing

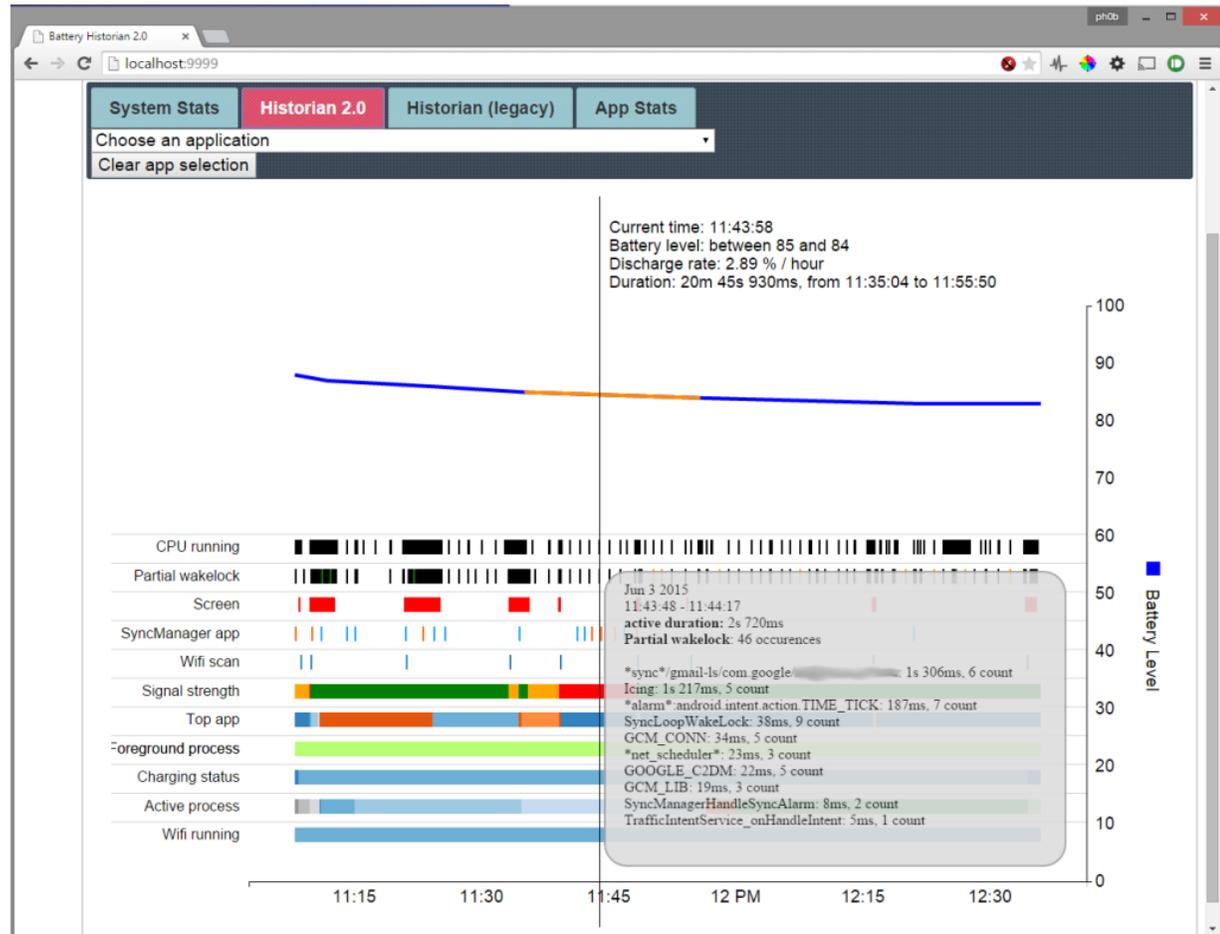


Starting Battery Historian 2.0

- Generate a bug report using command prompt
 - `adb bugreport > batterystats.txt`
 - It will take few minutes if your session is long
- Upload the generated report to the Battery historian 2.0 localhost page

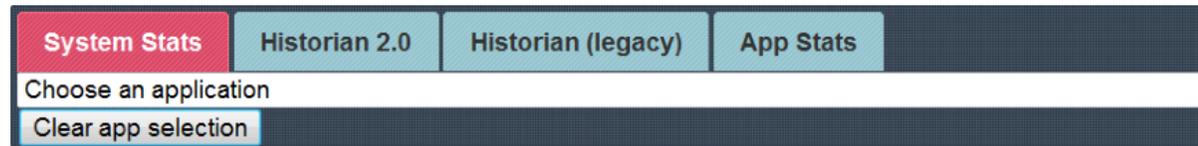
Report – Battery Historian 2.0

- It is a sample report generated by Battery Historian using the bugreport
- On the top is four tabs with various function including System stats, Historian 2.0 graph and App stats



Report – Battery Historian 2.0

- System Stats tab
 - Shows the general usage information of the device
 - E.g. Screen on/off time, Wifi usage, Session time



The screenshot shows the top navigation bar of the Battery Historian 2.0 application. The 'System Stats' tab is highlighted in red. Other tabs include 'Historian 2.0', 'Historian (legacy)', and 'App Stats'. Below the tabs, there is a text input field labeled 'Choose an application' and a 'Clear app selection' button.

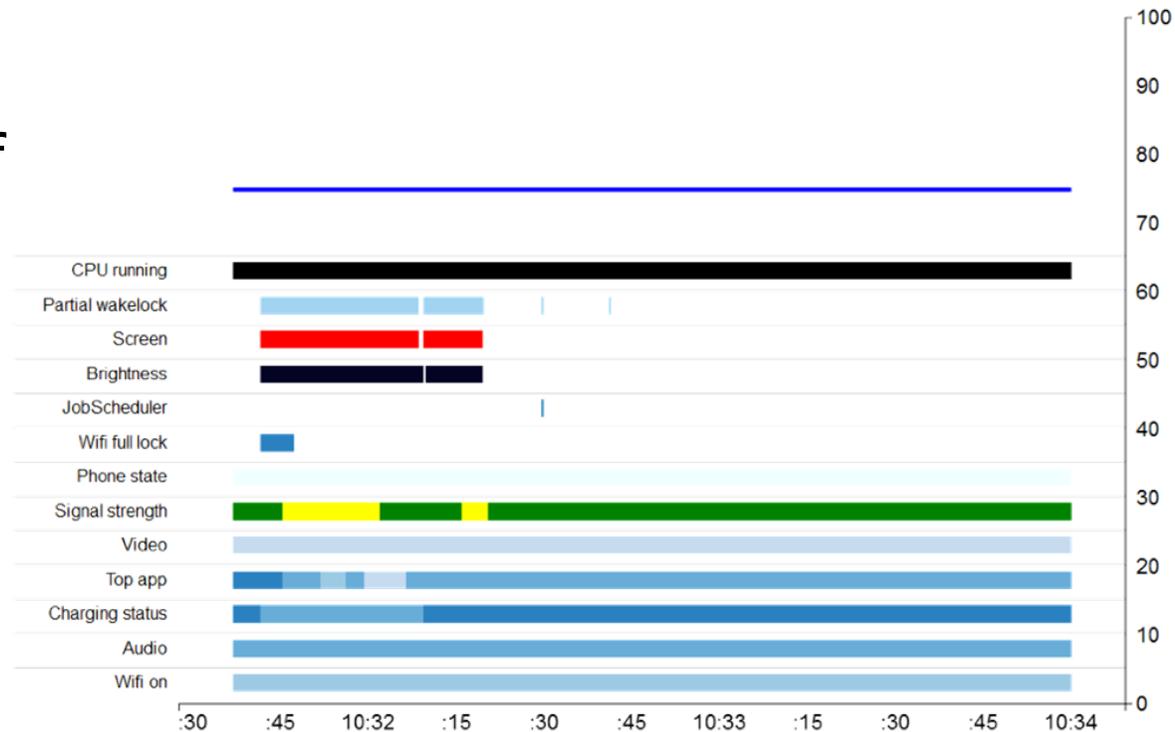
SM-G9200 LMY47X

Aggregated Stats:

Metric	Value
Device	SM-G9200
Build	LMY47X
Duration / Realtime	27.948s
Screen Off Discharge Rate (%/hr)	0.00 (Discharged: 0%)
Screen On Discharge Rate (%/hr)	0.00 (Discharged: 0%)
Screen On Time	27.1s
Screen Off Uptime	848ms
Userspace Wakelock Time	319ms
Kernel Overhead Time	529ms
Mobile KBs/hr	0.00
WiFi KBs/hr	0.00
Mobile Active Time	0
Signal Scanning Time	0

Report – Battery Historian 2.0

- Historian 2.0 tab
 - Shows a details graph of battery usage
 - Power usage on each application/ process is listed from more to less
 - Y axis shows the battery level while X axis shows the time line



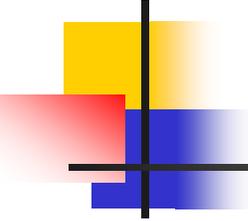
Report – Battery Historian 2.0

■ App Stats tab

- Shows the detailed power consumption of each application/process
- Individual application/process can be selected for viewing the statistics

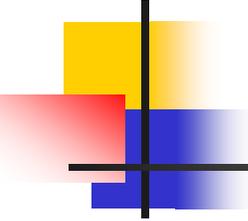
The screenshot displays the 'App Stats' tab in Battery Historian 2.0. The interface shows the selected application 'ANDROID_SYSTEM (Uid: 1000)' and a 'Clear app selection' button. Below this, a table lists application details: Application (ANDROID_SYSTEM), Version Code, UID (1000), Computed power drain (0.01%), and Foreground (1 times over 1s 416ms). There are also sections for 'Contained apps:', 'Wakelocks:', 'Services:', and 'Processes:' with 'Show' and 'Hide' buttons. At the bottom, a table lists processes with columns for Process Name, User Time, System Time, Foreground Time, # Starts, # ANRs, and # Crashes.

Process Name	User Time	System Time	Foreground Time	# Starts	# ANRs	# Crashes
system	6s 280ms	3s 110ms	0ms			
surfaceflinger	1s 770ms	2s 580ms	0ms			
servicemanager	290ms	350ms	0ms			
com.android.settings	150ms	50ms	50ms			
com.samsung.inputmethod	70ms	0ms	0ms			
com.sec.android.emergencymode.service	60ms	70ms	0ms			
argosd	10ms	50ms	0ms			
mcDriverDaemon	0ms	20ms	0ms			



Exercise #8 (25 mins)

- Test the Estock app
- Try to generate a bug report using adb
- Upload the generated report to the Battery historian 2.0 to see the statistics
- Find out which process(WIFI, etc.) uses up most of the battery when testing the application



Thank you

- *Your feedback is important to us.
Please fill in the evaluation form.*