ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (357-360), Month: July - September 2014, Available at: www.researchpublish.com

Android Based Network Analyzer

[1]Pooja Ranga, [2]Prof. Mr.Kapil

[1, 2] Department of Computer Science, South Point College of Engineering, Sonipat, Haryana, India

Abstract: **Android is an open source and Linux -Based Operating System mobile devices such a smartphones and a tablet computer. Android was developed by open hand set alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their application should be able to run on different devices powered by Android .The first beta version of the Android software development kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0,was released September 2008. Android is a powerful operating system competing with Apple4GS and support great features. Android applications are usually developed in the java language using the Android Software Development kit. In this we create a simple android based network analyzer which can give the user statistics about the network in a simple, user friendly interface .**

Keywords: **Android; linux based operating system; alliance; Apple4GS.**

I. INTRODUCTION

The main objective of this project is to create a simple android based network analyzer which can give the user statistics about the network in a simple, user friendly interface. The number one question network operators hear today is "my application is slow, what's wrong with the network?" To answer this question, the general solution is to install some specialized packet sniffing hardware/software, spend lots of time gathering data, and then find someone with the engineering expertise required to analyse the application's traffic. This complexity means that most performance problems, as opposed to connectivity problems, remain an unsolved mystery. To address this problem, we have developed an android based "Network Analyzer Tool" application, is based on a custom NDT [6] api. NDT stands for Network Diagnostic Tool which uses a Web100 [1] based approach to perform the basic network analysing functions. The Network Analyzer Tool uses a scheduler and a database engine to schedule tests on the mobile phone over the NDT api. The scheduler gives the user an ease to schedule the tests at required times. The results from the tests can be exported to a csv/text file for use in further analysis by the use.

The Network Diagnostic Tool (NDT)

Configuration and performance testing to a user's computer. It uses a well-defined protocol (Network Diagnostic Protocol NDTP) to provide a reliable communication link between client and server and has a protocol documentation which is sufficient to allow outside parties to write compatible NDT clients without consulting the NDT source code. Several studies have shown that the majority of network performance problems occur in or near the users' desktop/laptop computer. These problems include, but are not limited to, duplex mismatch conditions on Ethernet/Fast Ethernet links, incorrectly set TCP buffers in the local network infrastructure (e.g. first hop switch/router, hubs, etc.) and bad or dirty cables. NDT seeks to determine why a network connection exhibits certain performance characteristics. In addition to measuring the familiar The Network Diagnostic Tool (NDT) is a client/server program that provides network upload and download speeds of a user's connection, NDT also performs tests that can assess factors such as latency, packet loss, congestion, bad cables, out of order delivery and bottlenecks on the end-to-end path from client to server (on either side of the connection).

NDT Design and Working

Network Diagnostic Tool (NDT) uses a Web100 [WEB100] based approach to perform Network analysis. The basic premise is that by combining the Web100 data with measured TCP throughput data NDT can identify:

1. When normal network congestion limits throughput
2. When a network link is set to half duplex
3. What the bottleneck link is (Dial-up to 10 Gigabit Ethernet)
4. That a duplex mismatch condition exists
5. That a faulty cable or NIC is corrupting packets.

Forks a child process to handle the test and returns to a listen state. If a test is already in NDT app contacts the server application and requests a new test begin. The server application progress the child process fails and the client is notified that a test cannot begin yet. The user must start again to request a new test. Assuming that no other test is in progress, the child process initializes successfully and the server puts two (2) new TCP sockets into the listen state. The server then notifies the client that testing can begin. The client starts by opening and closing a connection on one (1) of the listening sockets. This test performs a simple NAT detection test by having the server return the socket peer data associated with this connection. Then it performs middlebox test.

After this test completes the server informs the client that it can begin a data streaming throughput test. The client opens a new connection to one (1) of the listening server sockets and streams data to the server for ten (10) seconds. At the completion of this test, the server and client save the test results. At this point it is possible to retrieve some KIS (Kernel Instruction Set) variable data from the server. Unfortunately, the data collected at this time has not proved to be very useful. The problem is not that the data is missing, but that a receiving TCP does not record interesting data. For example, a receiving TCP does not calculate an RTT value for the connection. The receiver side KIS variables are still being reviewed to determine if useful data can be extracted for diagnostic purposes. After this throughput test completes, the server informs the client to begin a second throughput test. The client connects to the second listening socket and the server streams data to the client for ten (10) second. At the completion of this test, the KIS variables for this connection are retrieved and transmitted back to the client. The server process then runs through the detection algorithms, described below, and writes the results out to a log file. This log was used to generate the information about the network. We described the detection algorithm using by NDT

Full or Half Duplex link:

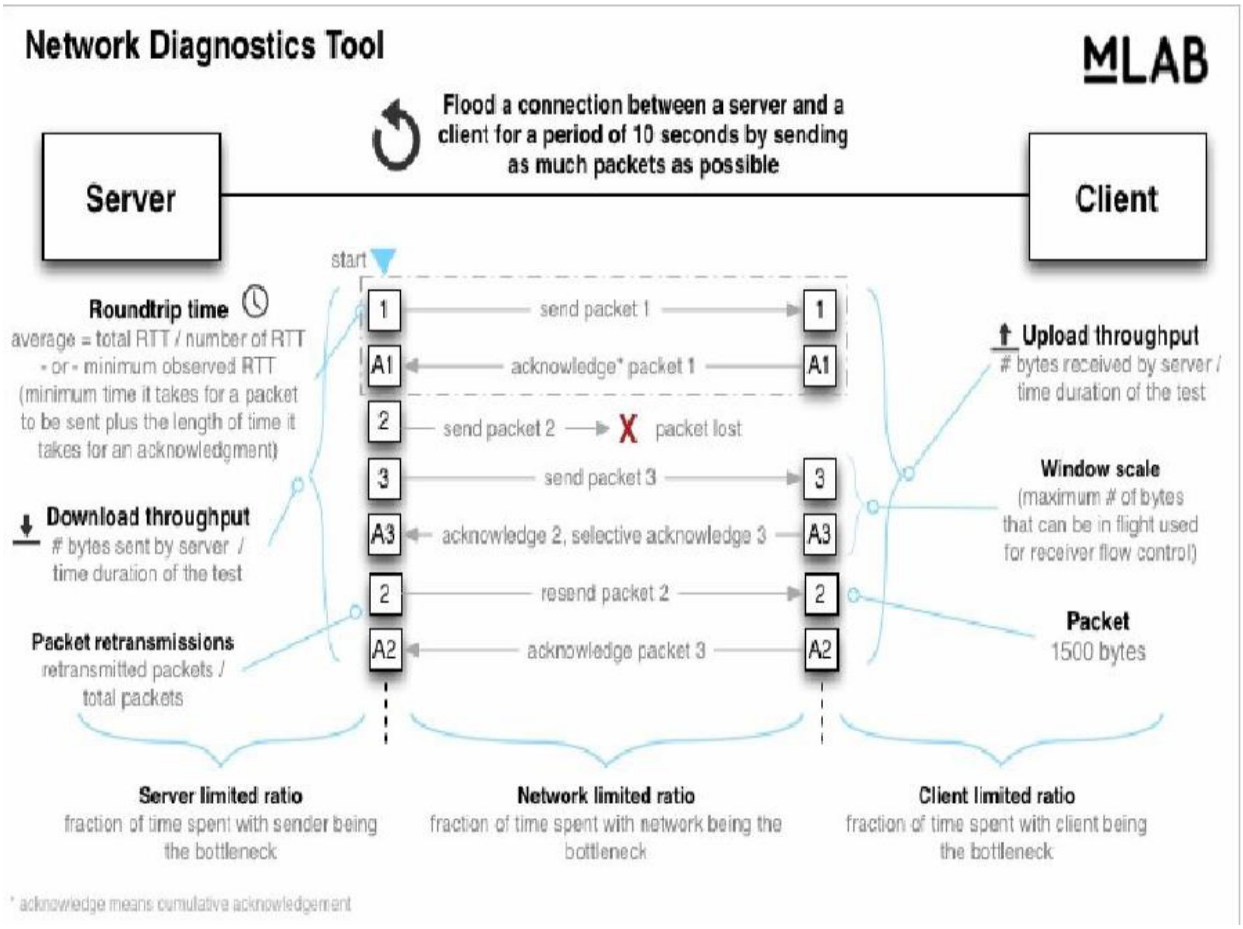
The NDT identifies half-duplex conditions by a logical AND of the following 4 tests:

1. The connection is receiver buffer limited over 95% of the time.
2. The connection transitioned into the receiver buffer limited state over 30 times per second.
3. The connection transitioned into the send buffer limited state over 30 times per second.
4. The link detection algorithm reported the bottleneck link is Fast Ethernet.

The NDT reports congestion by a logical AND of the following 4 tests:

1. The packet loss rate is less than 1%. A properly operating TCP stream will see very little packet loss even under congestion. This is due to the congestion control algorithms that cause TCP to lower the congestion window which limits the amount of data the sender can inject into the network.
2. The connection is network congestion limited more than 20% of the time.
3. A duplex-mismatch condition has not been detected.
4. The connection entered the TCP slow-start state.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (357-360), Month: July - September 2014, Available at: www.researchpublish.com



II. IMPLIMENTATION

Network Analyzer Design

The Network Analyzer Application is divided in 5 basic modules. The analyzer, the API, the Database Engine, the Background service and the Export engine. Each module is described in the following subsections.

The Analyzer

This is the main entry point of the application, it contain main home screen Activity. Home screen activity consists of *List Fragment* and *Fragment* for the detail view. This home screen is capable of displaying both mobile and tablet layouts. This home activity is responsible for connecting all functionalities of application like running network analysis test (NDT API tool kit), database engine, android background service, application scheduler and export engine. This contains list buttons for running test, create test bench and showing results.

Run test

This detail Fragment contains layout for running network analysis test, and display results of the test back on the screen.

Create Scheduler

Using this Fragment we are able to create test bench by providing duration of test bench in minutes or hours or days and interval between two consecutive test in seconds or minutes or hours. This scheduler is used run test by service in ackground and results are stored in the database for later reference.

Show results

Shows the list of results of test bench in summary as list button label and we can download detail result report as a csv file by click on respective list button.

III. RESULT AND CONCLUSION

The testing of the application showed that the Network Analyzer Application made it easier for a mobileuser to analyze the mobile network. The problems faced during the application development were:

Problems:

1. The application would crash at odd times due to what seemed to be a problem between the integration of the API and the database engine.
2. During some tests, the scheduler would either crash on submitting the schedules or would not send the information to the database engine, failing which would let the scheduling of the api via the service fail.
3. All the bugs and performance improvements are being looked into as this report is being written.

Albeit all the afore mentioned problems, the integration of NDT api was a success. Users can change the parameters of the tables to focus on what analysis of the network they are interested in. NDT offers almost 25 values for the network which in turn provides the user with a flexibility to analyze the network in an in depth manner.

REFERENCES

[1] http://www.web100.org/

[2] http://www.vogella.com/tutorials/AndroidSOLite/article.html

[3] http://www.vogella.com/tutorials/AndroidServices/article.html

[4] http://code.google.com/p/m-lab/wiki/PDEChartsNDT

[5] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.5110&rep=rep1&type=pdf

[6] http://ndt.anl.gov/toolkit/