行政院國家科學委員會專題研究計畫 成果報告

嵌入式網路通訊裝置評比技術與工具之研發(中心分項)--子計畫一:嵌入式網路通訊裝置應用效能評比技術與工具之研發

研究成果報告(精簡版)

計 畫 類 別 ： 整合型
計 畫 編 號 ： NSC 97-2218-E-009-032-
執 行 期 間 ： 97 年 08 月 01 日至 98 年 07 月 31 日
執 行 單 位 ： 國立交通大學資訊工程學系（所）

計 畫 主 持 人 ： 林盈達

處 理 方 式 ： 本計畫可公開查詢

中 華 民 國 98 年 10 月 19 日

行政院國家科學委員會補助專題研究計畫 ■成果報告
□期中進度報告

嵌入式網路通訊裝置測試中心：
方法與工具之實驗、開發與推廣

計畫類別：□個別型計畫　　■整合型計畫
計畫編號：NSC 97-2218-E-009-032
執行期間：97 年 08 月 01 日至 98 年 07 月 31 日

計畫主持人：林盈達 教授
共同參與人員：陳一瑋、江易達、許郡泓

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立交通大學

中　華　民　國　98　年　8　月　1　日

行政院國家科學委員會專題研究計畫成果報告

計畫名稱：嵌入式網路通訊裝置測試中心：方法與工具之實驗、開發與推廣

子計畫一：嵌入式網路通訊裝置應用效能評比技術與工具之研發

計畫編號：97-2218-E-009-032

執行期限：97 年08 月01 日至98 年07 月31 日

主持人：林盈達 教授　國立交通大學 資訊工程學系(所)

1. 摘要

嵌入式系統平台已被大量應用在商用及家用產品上如 PDA 和 WiFi Phone，但是對於嵌入式系統網路裝置在使用者端的應用效能上卻缺乏可用的測試工具，廠商在開發產品之後無法精確得到其效能，當產品有關於效能方面的問題也無從得知可以改善的方式，為此，開發「嵌入式系統網路通訊裝置應用效能評比工具」，除了要提供黑箱測試數據外，更應設計對於開發者可進行加強以及偵錯的灰箱及白箱測試，協助開發者清楚了解產品在使用者端所遇到的缺點以及相關改進的方法。

本計畫開發的項目，是針對目前熱門的嵌入式系統網路通訊產品，進行相關網路應用服務效能評比(如: Throughput, Session Capacity, Session Rate, Voice Quality)，利用NBL現有的工具以及Open Source軟體，以Repeatable、Scalable、Automatic、Integrated等四個條件為前提，進行**1. 測試計畫及方法之研發**, **2. 對測試設備無依賴性之測試工具研發**, **3. 整合型測試工具研發**，配合總計畫提供的共通測試平台，提供各種不同層級的背景流量，更能貼近實際上使用者端使用的環境，在測試工具的開發規劃裡，預計進行兩項測試工具的開發，其中包括：連線量測試工具、SSL VPN連線量測試工具以及語音品質整合測試工具，這些都是目前所所缺乏的應用效能測試工。達成對嵌入式網路通訊裝置進行完整效能的測試及評比，除了能幫助產品開發者進行效能改進及選擇適合使用的元件平台，亦可做為學術上進行相關研究時有實際上產品的數據供做先進技術的研發，並將此系統原始碼公開，提供開放原始碼社群使用及研究。

關鍵詞： **嵌入式系統、網路通訊裝置、網路應用服務效能、開放原始碼**

2. 緣由與目的

晶片設計以及製程技術的進步，晶片組以及其相關整合產品效能不斷的提升且價格更趨便宜，使得許多平常可見的商用或家用產品皆可利用嵌入式系統實現之，尤其現在更強調這些產品的網路連通性，以及彼此之間與個人電腦和網際網路的整合性，網路功能不再是個人電腦的專利，而是各項資訊產品、家電等各種產品用來整合個人和公司資訊及各項功能的方案；然而針對這類型的整合性產品(ex: PDA, WiFi Phone)卻缺乏可以進行嵌入式網路通訊系統效能評比的方式，針對單一元件可以直接進行該元件測試即可達成，然而對 System Vendor 而言卻非如此，Chip Vendor可以經由單一晶片測試達到其所需求的結果，但是 System Vendor 卻必須將目標放在產品的系統效能上，除此之外現在 Chip Vendor 已漸漸走向 Total Solution 的模式，也就是說 Chip Vendor 要自己來開發 System，因此更會期待一個測試中心可以對整體系統的效能表現作一個完整的測試，所以子計畫一的目標應考慮整合好的產品所必須要了解的系統效能，並且要能經由不同的應用程式測試，達成對整體系統的分析，根據不同的嵌入式網路通訊系統(註一)所欲得之結果(註二)，開發針對不同型態(註三)的測試項目。

註一: handheld、server、network。

註二: Throughput、Latency、Bandwidth Consumption、Response time、TCP connection rate/capacity IPSec tunnel rate/capacity、Voice Quality、SIP call rate/capacity、WLAN association rate/capacity。

註三: Black, Grey, White Box。

Black box 測試：把待測物視為一個黑盒子，餵入合成的存取樣式，從外部測量基本的效能的讀數，如 Throughput。這種測試可以測出在通用或是已知的特殊情況的行為及效能。

Gray box 測試：不同的網路通訊裝置有其特殊的使用機制，如 NAT/Firewall 上設定的規則，必須進入 Gray Box 階段。並透過廠商提供的控制介面，進一步的從待測物上取得 black box 讀取不到的數據，如：Default Rules 和 Matching priority，進行 Throughput 和 Latency 效能評估。

White box 測試：主要用來取得更詳盡的測試數據與事件截取，因為有了原始碼，則可以直接針對該演算法取得所有與效能相關的數據。

而評斷嵌入式系統效能的好壞可以由兩個方向進行區分，一者為硬體上的效能，指的是裝置上的CPU、Memory和I/O等裝置的效能，例如CPU的處理速度為一秒鐘可以執行多少指令集，記憶體容量有多少、能以多快的速度接受存取，I/O在複製、搬移等動作時，每秒可以存取多少容量；另一者為網路系統效能，如Throughput、Latency、Loss和Session Capacity等，Throughput是對於網路系統最直接的評斷方式，直接指出該網路系統在不造成封包丟棄的狀態下，能處理的最大數量封包，Latency則是封包從進入該網路系統至離開該網路系統時，在系統中停留的時間，即為系統所造成之延遲，Loss則是當輸入loading超過系統所能負荷的最大負載時，所造成的封包遺失為多少，Session Capacity則是在測試針對某單一通訊協定(如TCP)，網路系統能承受的連線數量為多少；本子計畫將針對嵌入式系統網路裝置效能部份進行應用效能評比工具開發。

3. 研究內容

本計畫研究所開發之工具為SSL VPN Tunnel Tester (SSLTC)與Integrated Voice Quality Test(IVQT)，其詳細架構如下。

(1) SSLTC

用戶端可以利用VPN方式連線，進行私密安全連線，以防止資料在網路上進行傳輸時，遭到駭客竊取，但是加密連線必須使用大量的運算，以及過濾，使得該類型的網路安全產品可容納的VPN容量無法達到實際需求，考量在大部份的情況下，利用VPN進行連線時，許多的連線都是在進行網頁瀏覽，因此並非需要在Client和Server端之間保持建立連線的狀況態，也可由此減輕VPN網通裝置的負擔，利用SSL VPN的方式進行資料的流通，以Linux 為SSL VPN的client，產生同時多條的SSL VPN tunnels (Full Tunnel Mode)與users來連上SSL VPN server，可用來自動化測試tunnel capacity、user capacity、stability。

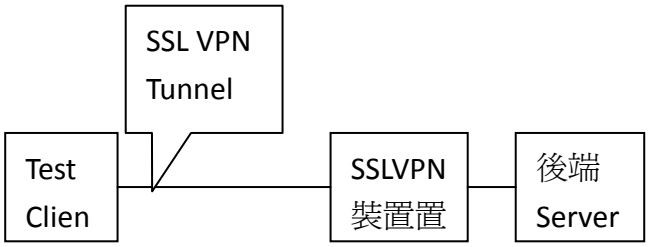
本系統的軟體架構主要是由幾項元件所構成: Traffic Generator – SSL VPN Tunnel, Traffic Generator – Background Traffic, Controller，以下說明各個元件所扮演的角色以及如何進行測試與蒐集數據:

Traffic Generator – SSL VPN Tunnel： 使用一台 PC 製造出同時上千條的 SSL VPN tunnels。

Traffic Generator – Background Traffic：除了建立 SSL VPN tunnel 外，也可產生網路流量導入 SSL VPN tunnel 中，如 HTTP、FTP.. 等等。

Controller：會使用一台 PC 作為 controller，controller 的功能一是用來 trigger traffic generators，二是用來抓取 DUT 上的 information 來判斷該次測試的結果是 Pass or Fail。

SSLTC 之系統架構如下所示：



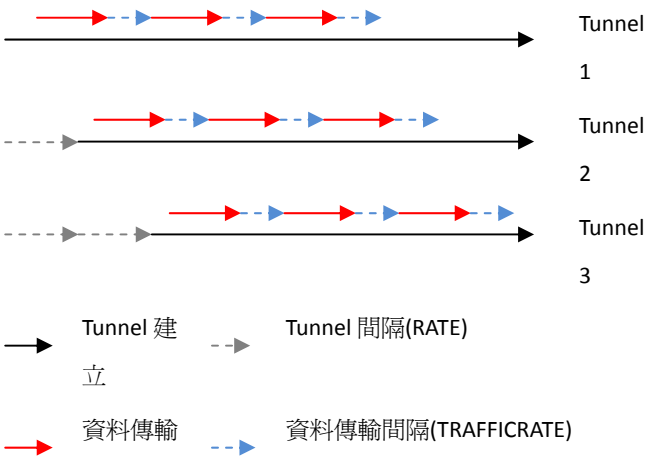
在 Client 上先執行 pptpd，然後執行測試程式。測試程式會先登入 SSLVPN 裝置，認證成功後由 pptpd 呼叫 pppd 建立 ppp 介面，

連到後端 server 所在的 subnet，之後再由測試程式透過建立好的 SSLVPN Tunnel 向後端 server 抓取資料，驗證是否成功連線。

目前支援 capacity 及 throughput 兩種測試模式，capacity 會不斷建立 tunnel，直到指定的數量或是所有帳號都用完為止，每個 tunnel 建立的間隔時間為 RATE。每個 tunnel 建立後，會依據指定的 TRAFFICRATE 時間間隔，向後端 server 抓取資料，驗證 tunnel 是否正常。

Throughput 測試會先建立指定數量的 tunnel。等全部建立好後同時向後端 server 抓取資料。每條 tunnel 會各自計算 throughput(總傳輸的資料量/(結束時間-開始時間))，另外也會算總共的 throughput，以全部 tunnel 的傳輸量除以(所有傳輸完成時間-開始傳輸時間)。

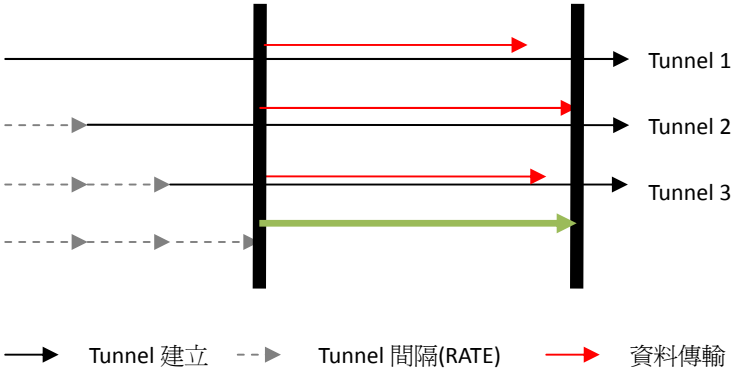
目前支援 capacity 及 throughput 兩種測試模式，capacity 會不斷建立 tunnel，直到指定的數量或是所有帳號都用完為止，每個 tunnel 建立的間隔時間為 RATE。每個 tunnel 建立後，會依據指定的 TRAFFICRATE 時間間隔，向後端 server 抓取資料，驗證 tunnel 是否正常。



Capacity 測試

Throughput 測試會先建立指定數量的 tunnel。等全部建立好後同時向後端 server 抓取資料。每條 tunnel 會各自計算 throughput(總傳輸的資料量/(結束時間-開始時間))，另外也會算總共的 throughput，以全部 tunnel 的傳輸量除以(所有傳輸完成時間-開始傳輸時間)，時間計算如下圖的 ➡ 部分。

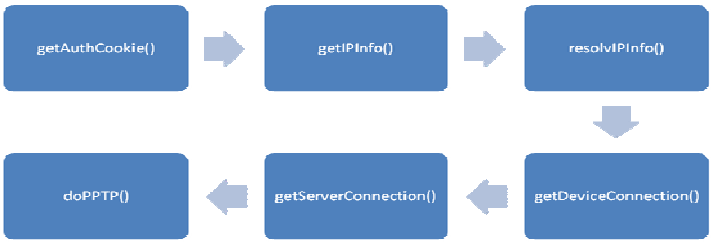
因為 smb 是以 jcifs 還是庫來連線，計算 smb 流量時溝通的流量並不會計入。所以時間是以抓資料的時間來計算。http 和 ftp 流量與時間均會把溝通的部份算進去。



Throughput 測試

程式首先會讀取命令列參數，取得設定檔名與要測試的 tunnel 數。之後讀入設定，如果不是前面列的參數(如 TRAFFIC、RATE)，就建立一個 test instance，傳入帳號密碼及其他資訊，若密碼後有接 IP(以,分隔)，則使用這組帳號密碼連線時，會連到此 SSLVPN 裝置，而不是 DEVICE 指定的 IP。全部建好後，就依設定的間隔執行。每一個 instance 用一個 thread 跑，所以測試時會有很多個 thread 同時跑。

Tunnel 建立流程如下:



(1) getAuthCookie()
連到 SSLVPN 裝置的 https 網頁，以給定的帳號密碼登入後取得 cookie

(2) getIPInfo()
用第一步取得的 cookie，一樣連到 SSLVPN 的 HTTPS，送出
CONNECT 127.0.0.1: 10443

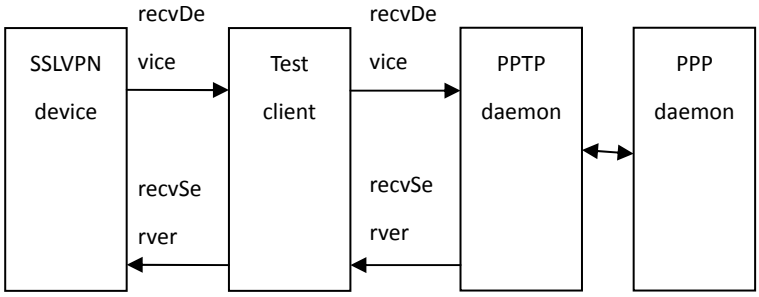
HTTP/1.1\r\n
Host: 127.0.0.1\r\n
Cookie: authtok= xxxxxxx\r\n\r\n
若回傳 HTTP/1.1 200，代表成功，之後傳送如下，抓取 IP 設定資訊。
GET /L2TunCFG xxxxxxx

(3) resolvIPInfo()
分割第二步抓到的資訊，取得 interface IP 等資訊

(4) getDeviceConnection()
同樣連到 SSLVPN HTTPS，送出：
Get /L2TunREADY xxxxxxx
之後此連線要持續，整個 SSLVPN Session 中不可中斷。

(5) getServerConnection()
與本機的 pptpd 溝通，這裡會先額外送一個含有第三步 IP 資訊的封包，以及本地的 GRE port，交給 pptpd 的 getPreHeader()處理。Pptpd 呼叫 pppd 建立 interface 後，會呼叫 extra/etc/ppp/ip-up 建立 routing entry，ppp interface 關閉時也會呼叫 extra/etc/ppp/ip-down 刪除 routing entry。

(6) doPPTP()
分成兩步同時執行。首先是 RecvServer，建立一個 thread，由 test.java 中的 recvServer()函式處理。從 pptpd 抓取封包送至 SSLVPN 裝置。
另外是 RecvDevice，由 test.java 中的 recvDevice()處理。從第四步的連線抓取封包，去除 header 後寫至第五步的 pptpd socket 建立後取得的 gre socket。



流量測試的程式，依據設定檔中的 TRAFFIC 設定，呼叫 HttpClient、FtpClient、SmbClient 執行。

而統計流量時，每個*Client 會各自計算總共收到的 byte 數，再除以時間。另外 test client 會傳進一個 AtomicLong，讓每一個 client 將他收到的 byte 數也加進去，以計算總共的 throughput。

而在 SmbClient 部分，因為是使用 jcifs library 處理，沒辦法抓到一開始溝通，登入部份的封包，所以只計算下載資料的封包。時間上會另外計算溝通所佔的時間，計算 throughput 會扣掉所有這段時間的平均。

(2) IVQT

PDA或是WiFi Phone可以透過有線及無線網路進行VoIP連線，利用SIP協定進行撥打電話的動作，再藉由直接或間接使用RTP協定封包進行語音封包傳送，利用有線的方式進行VoIP連線不必考慮移動性的問題，但是在無線網路環境下則不同，除了無線網路使用的MAC Layer協定(主要是CSMA/CA)限制了VoWiFi(Voice over WiFi)使用上無法充份使用預定的頻寬，實際使用上還有關於在無線網路下Roaming等問題，所以在有線及無線網路的環境下，皆可利用各自條件不同的變因，使用PESQ測試語音品質的標準，測試對於待測物品的語音通話品質影響。

本系統的軟硬體架構主要是由幾項元件所構成: NIST-Net Controller，Background Traffic Generator，Azimuth Platform，Integrated Controller，以下說明各個元件所扮演的角色以及如何進行測試與蒐集數據:

NIST-Net Controller：需能支援使用

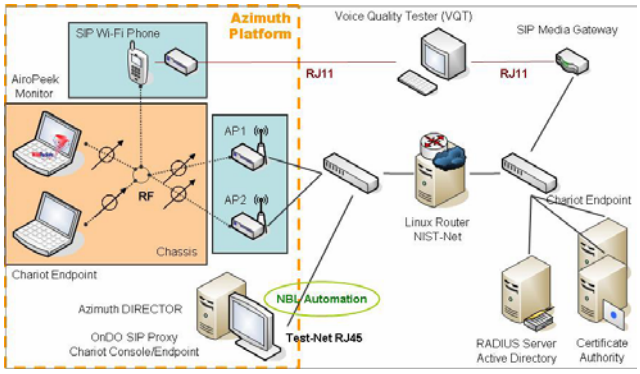
NIST-Net 控制對於網路環境的影響，如在收話端與受話端之間 Traffic 受環境影響的控制，封包延遲或 Loss 等條件，用以測試在該環境下對於語音通話品質的影響。

Background Traffic Generator：需能支援在不同型態的Background Traffic之下，擷取語音相關的封包進行語音品質測試，Background Traffic可以為HTTP,FTP, or P2P etc.，藉以觀察在不同的traffic以及不同的loading之下對Voice Quality的影響。

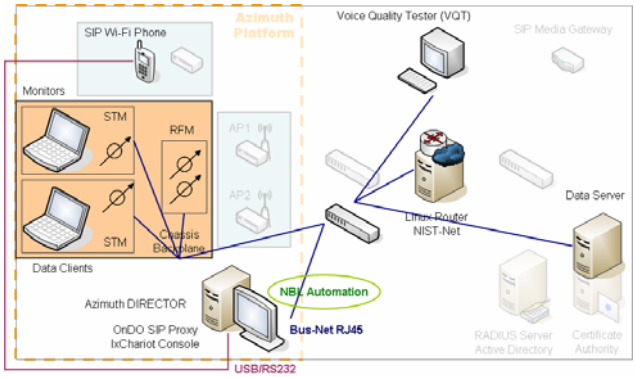
Azimuth Platform(For Wireless Only)：利用 Azimuth 平台進行 RF Attenuator 控制，可以擬似對於 WiFi Phone 的移動模式，藉以測試在移動狀態中的 WiFi Phone 對於 Voice Quality 的影響，藉由本測試可以分析關於 WiFi Phone 與 AP 在移動態狀下的適應性，由本測試可以進一步進行在移動中需進行 Roaming 時，WiFi Phone 對於 Roaming 機制的調適，選擇以何機制和時機進行 Roaming 的動作，以及在 Roaming 的同時對於語音品質的影響為何。

Integrated Controller：上面的測試都是只有針對單一的情況進行測試，但是這樣的測試無法進行多次的重複使用，每次進行測試時都要重新再設定不同的環境和不同的條件及劇本，做一次測試要花費許多的時間進行其它工具的設定，再者在動態設定的情況下，會造成各工具之間 Timing 的非同步，造成測試不精確，因此本項目支援的目的是要做到在測試時可以利用單一視窗介面，進行對 WiFi Phone, NIST-Net 和 Azimuth 同步的控制以及批次處理。

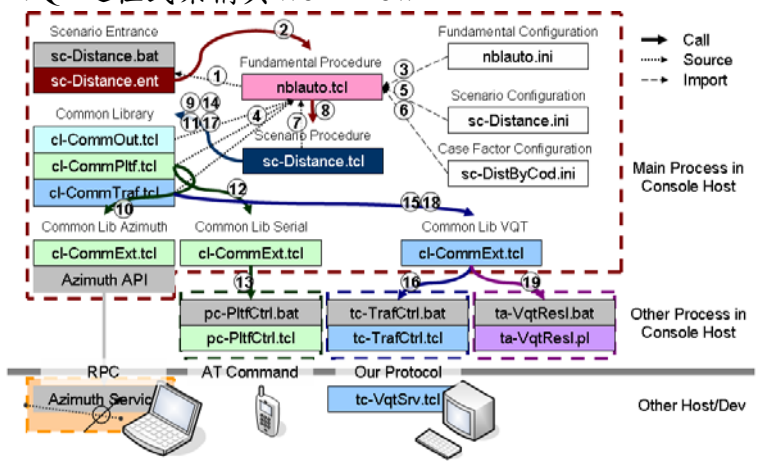
IVQT之整體系統架構如下所示：



IVQT之自動化控制架構：



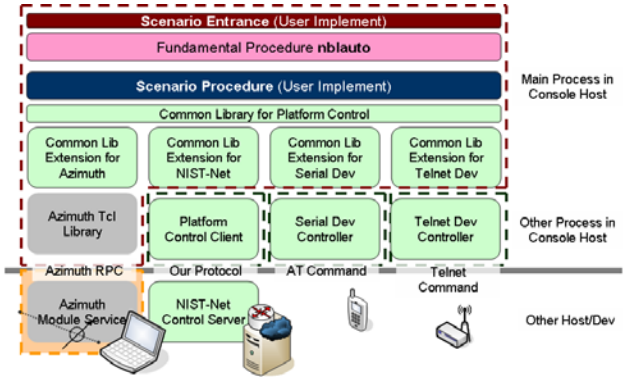
IVQT之程式架構與Work Flow：



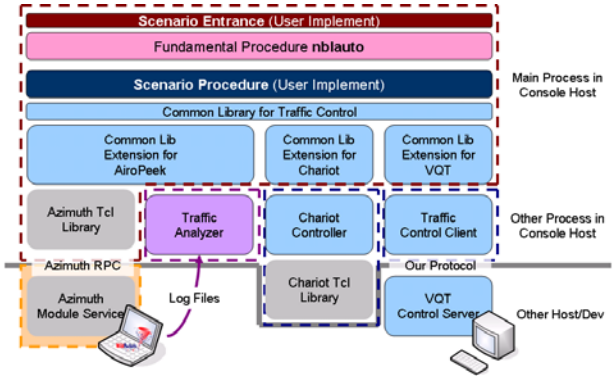
整個系統的軟體架構可以分成三部份：Platform Controller、Traffic Controller及Traffic Parser，初始時可以利用tcl進行對VoIP產品與AP間的距離，控制Azimuth的RF Module造成Voice Phone與AP之間有訊號上的衰減等因子，透過距離與RF訊號衰減的公式，進行RF訊號強度調整，同時套用入距離動態調變，並啓動相關的module(ex: 利用chariot產生背景流量等)，而整個系統啓動的同時，VQT亦開始進行voice quality的量測，包含語音訊號產生、語音訊號傳送與語音訊號品質量測等。最後會將測試的結果及過程，將相關的記錄

檔送至Traffic Parser進行後端的分析，以及語音量測分數產生，達成對語音訊號品質量測的結果及報告。

IVQT之Stack Diagram(Platform)：



IVQT之Stack Diagram(Traffic Control and Analysis)：



4. 研究成果與討論

本計畫已經能利用相關的工具以及已經開發的程式及模組，進行與SSL VPN Tunnel及Voice Quality量測的工具，其中SSL VPN Tunnel部份，由於目前SSL VPN Tunnel技術並沒有相關的標準製定，因此目前已經與多家知名大廠合作，廠商也願意提供產品在進行研發時所使用的protocol stack與本計畫合作，在本計畫本所使用的建立SSL VPN Tunnel技術有一定的透明度，使本計畫中研發來給SSL VPN相關技術所使用的測試計畫，不旦有黑箱測試，同時也有灰箱測試的部份，有助於本計畫團隊對於SSL VPN技術應用在學術用、商用平台上能精確的了解其Tunnel建立技術；而IVQT部份，由於涵蓋相當大範圍的不同技術(Voice Codec, Voice Generator, Voice Quality Tester, 802.11a/b/g, Background Traffic)及開發工具(Azimuth, Abacus, Net-NIST, IxChariot)的整合，雖然大部份的內容都是黑箱測試，但是對於各項通訊協定及語音編碼、語音量測等相關的標準的了解，有助於未來本團隊在進行相關工具開發及學術研究的延展性，而同時整合多項測試工具實行利用有限的空間，實施更大規模的測試，有助於未來在工具開發及執行相關測試專案時，能接受各種不同腳本進行有彈性的調整及高度客製化。

5. 參考文獻

[1] Tilman Wolf and Mark Franklin, “A Telecommunications Benchmark for Network. Processors,” Proc. of IEEE International Symposium on. Performance Analysis of Systems and Software, 2000.

[2] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “MiBench: A free, commercially representative embedded benchmark suite," IEEE International Workshop on Workload Characterization, pp. 3-14, 2001.

[3] S. Bradner, Bechmarking Terminology for Network Interconnection Devices.RFC1242, July 1991

[4] S Bradner, J McQuaid, Benchmarking Methodology for Network Interconnect Devices, RFC 2544, March 1999

[5] D. Newman, RFC 2647 - Benchmarking Terminology for Firewall Performance, August 1999

[6] Yunfei Wu, Stephan Wong, and Stamatis Vassiliadis, “Real-Time Network Processing: An Investigation,”

[7] Manohar Castelino and Frank Hady, “Tutorial on NPF's IPsec Forwarding Benchmark,” Network Processing Forum

[8] Lee Eng Kean, Sulaiman bin Mohd. Nor, “**A** Benchmarking Methodology for NPU-Based Stateful Firewall,” IEEE 2003

[9] R. Ramjee, J. Kurose, D. Towsley, H.

Schulzrinne, "Adaptive Playout Mechanisms for packetized Audio Applications in Wide Area Networks", Proceedings of the Conference on Computer Communications (IEEE Infocom), Toronto, Canada, 1994

[10] K. Fujimoto, “Adaptive Playout Buffer Algorithm for Enchancing Perceived Quality of Streaming Apllications,” Osaka University, Japan, February 2002

[11] Henning Schulzrinne, Sankaran Narayanan, Michael Doyle and Jonathan Lennox, "SIPstone - Benchmarking SIP Server Performance," April, 2002

[12] C. Lee, M. Potkonjak and H.Mangione-Smith,”MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems,” Micro-30, November 1997.

[13] SJ Shah, “Network benchmarking,” netprl.calpoly.edu, 1997

[14] Sandra Johnson, Gerrit Huizenga, Badari Pulavarty, Performance Tuning for Linux Servers,IBM Press, May 2005

[15] Netfilter: firewalling, NAT and packet mangling for Linux, http://www.netfilter.org

[16] Tcl, http://tcl.sourceforge.net/

[17] Netperf, http://www.netperf.org/netperf/

[18] Netio, http://www.nwlab.net/art/netio/netio.html

[19] ipbench, http://ipbench.sourceforge.net/

[20] EEMBC, http://www.embedded-computing.com

[21] NIST NET, http://www-x.antd.nist.gov/nistnet/

[22] Azimuth Systems, http://www.azimuthsystems.com/

[23] Spirent, http://www.spirentcom.com/

[24] Radcom, http://www.radcom.com

[25] NBL, http://www.nbl.org.tw