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--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(中心分項)(2/2)

研究成果報告(完整版)

計 畫 類 別 ： 整合型
計 畫 編 號 ： NSC 99-2220-E-009-044-
執 行 期 間 ： 99 年 08 月 01 日至 100 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 100 年 10 月 30 日

目錄 (Cotent)

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。

另外，在嵌入式系統上剖析使用情境的執行時間，會面臨三個主要的問題:第一、使用情境的執行會橫跨不同軟體層來呼叫不同元件，第二、軟體層皆是由一種以上的程式語言所構成，第三、系統的儲存空間有限。在此我們設計一個分段反覆插入剖析方法，可以在多種程式語言及多個軟體層(multi-langue and multi-layer)平台上找出執行時間瓶頸。由單一的模組開始，進而逐漸地加入不同軟體層的不同模組來剖析，以避免在剖析過程中產生大量不必要的資料節省儲存空間，最後再把不同軟體層執行時所輸出的資料合併分析，藉此找出執行時間瓶頸。

3. 研究內容

本計畫研究所開發之工具為 SSL VPN Tunnel Tester (SSLTC)、Integrated Voice Quality Test (IVQT)，以及 Multi-Level Performance Profiling (MLPP)其詳細架構如下：

■ 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：

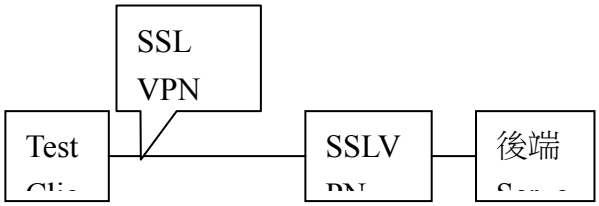
Traffic Generator – SSL VPN Tunnel： 使用一台 PC 製造出同時上千條的 SSL VPN

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：

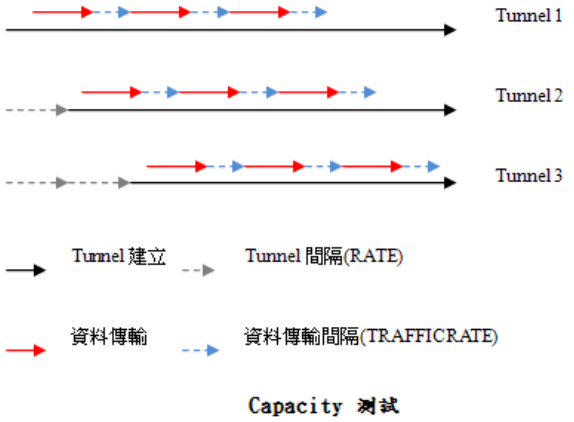


在 Client 上先執行 pptpd，然後執行測試程式。測試程式會先登入 SSLVPN 裝置，認證成功後由 pptpd 呼叫 pppd 建立 ppp 介面，連到後端 server 所在的 subnet，之後再由測試程式透過建立好的 SSLVPN Tunnel 向後端 server 抓取資料，驗證是否成功連線。

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。

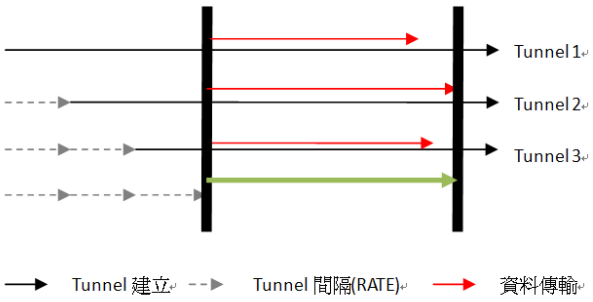
Throughput 測試會先建立指定數量的 tunnel。等全部建立好後同時向後端 server 抓取資料。每條 tunnel 會各自計算 throughput(總傳輸的資料量/(結束時間-開始時間))，另外也會算總共的 throughput，以全部 tunnel 的傳輸量除以(所有傳輸完成時間-開始傳輸時間)。

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

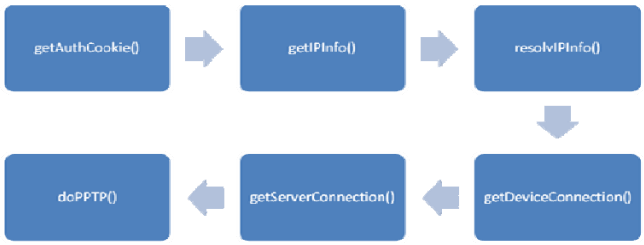
Throughput 測試會先建立指定數量的 tunnel。等全部建立好後同時向後端 server 抓取資料。每條 tunnel 會各自計算 throughput(總傳輸資料量/(結束時間-開始時間))，另外也會算總共的 throughput，以全部 tunnel 的傳輸量除以(所有傳輸完成時間-開始傳輸時間)，時間計算如下圖的 ➡ 部分。



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 tunnel 數，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

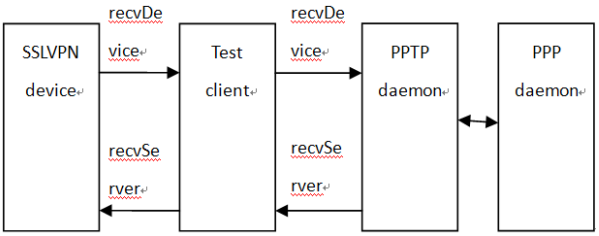
之後此連線要持續，整個 SSLVPN Session 中不可中斷。

(5) getServerConnection

與本機的 pptpd 溝通後會先額外送一個含有第三步 IP 資訊的封包，以及本地 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。

■ 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:

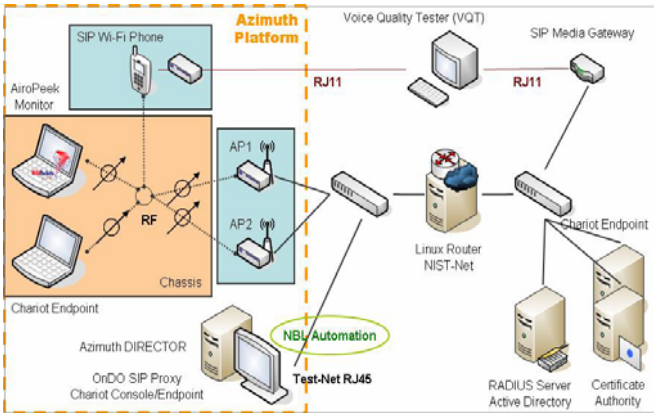
NIST-Net Controller：需能支援使用 NIST-Net 控制對於網路環境的影響，如在收話端與受話端之間 Traffic 受環境影響的控制，封包延遲或 Loss 等條件，用以測試在該環境下對於語音通話品質的影響。

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。

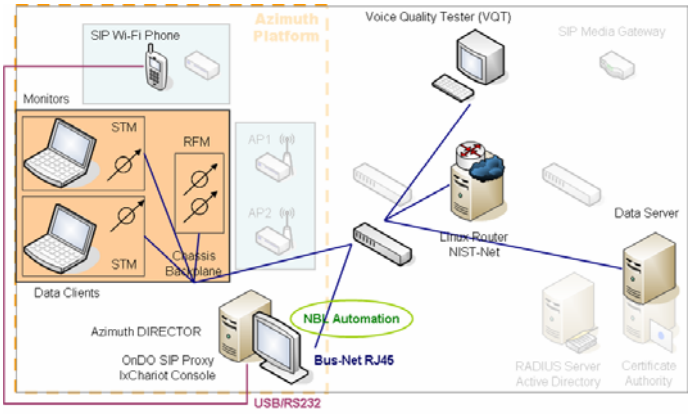
Azimuth Platform(For Wireless Only)：利用 Azimuth 平台進行 RF Attenuator 控制，可以擬似對於 WiFi Phone 的移動模式，藉以測試在移動狀態中的 WiFi Phone 對於 Voice Quality 的影響，藉由本測試可以分析關於 WiFi Phone 與 AP 在移動態狀下的適應性，由本測試可以進一步進行在移動中需進行 Roaming 時，WiFi Phone 對於 Roaming 機制的調適，選擇以何機制和時機進行 Roaming 的動作，以及在 Roaming 的同時對於語音品質的影響為何。

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。

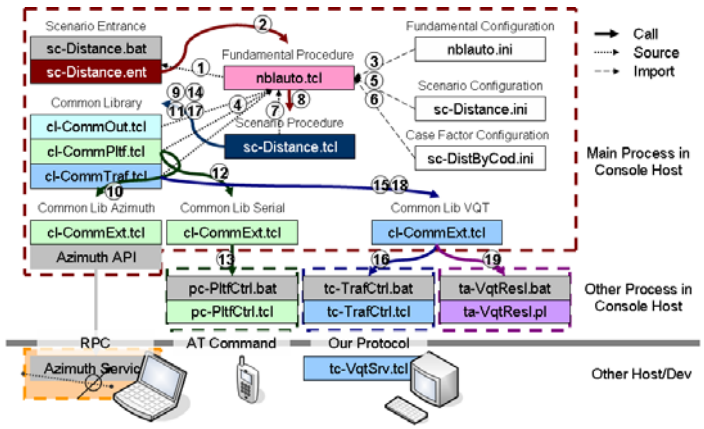
IVQT之整體系統架構如下所示：



IVQT之自動化控制架構：



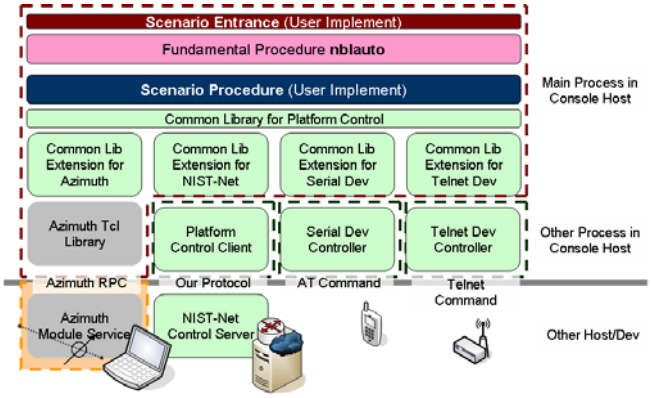
IVQT之程式架構與Work Flow：



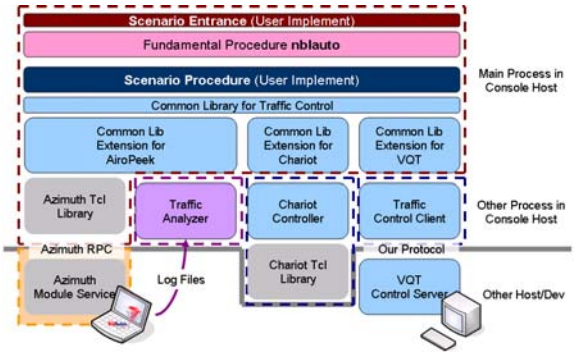
整個系統的軟體架構可以分成三部份：Platform Controller、Traffic Controller 及 Traffic Parser，初始時可以利用 tcl 進行對 VoIP 產品與 AP 間的距離，控制 Azimuth 的 RF Module 造成 Voice Phone 與 AP 之間有訊號上的衰減等因子，透過距離與 RF 訊號衰減的公式，進行 RF 訊號強度調整，同時套用入距離動態調變，並啓動相關 module(ex: 利用 chariot 產生背景流量等)，而整個系統

啓動同時，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)：



■ MLPP

在現今許多嵌入式系統有著以下特特：1）多層架構(multi-layer)，2）多種程式語言寫成(multi-language)，三）有限記錄檔(log)儲存空間。

析分析工具的選擇：在現有嵌入式系統評測工具中，若有支援多層分析的工具硬是能使分析複雜度大幅降低。其因一，分析工具學習時間縮短和原始碼修改的複雜度降低；其二，能夠使分析結果的格式簡化。本研究選用的 log 分析工具，皆是用於 user space 模組，不管位於那些層或者是什麼程式語言都能夠進行效能分析。至於對於 kernel 分析的工作，則交予 picks 和 printk 兩個函式，這兩者的使用方式與上述 log 工具的運作相當類似。

log 合併：在使用上述 log 分析工具和 printk 函式後，會發現其兩者的時間單位、日期格式以及環境基準是不同的。前者的時間單位為 μs，日期格式為從 cpu_clock 函式所輸出的 yy/mm/dd，且所有的值皆是與 CPU 綁在一起；後者時間單位為 ns，其日期格式是由 current_kernel_time 函式所輸出的一組數字，且所有數值皆是與 kernel 綁在一起。為了得到更精確的結果，我們會把所有分析工具的輸出統一轉換為 printk 函式的格式。

自動插 code：將分析所需的 code 插入原始碼是一個繁覆且枯燥的工作，所以我們實作了自動化 script，使得這些分析程式碼，能夠自動地被插入到指定的檔案或資料夾，亦或於分析完成後自動地被移除。這些自動化的 script 能夠運行在 C、C++和 Java 上，並能選擇輸出為 log 工具格式或 printk 函式的格式，使得整個效能分析的過程更加簡便。

多層架構與多種程式語言(multi-layer & multi-language)：在此研究中，我們同時運行多種分析工具模組，使之能應付多層架構與多種程式語言特性。然而，多種模組的使用上也衍生出新的問題──必需整合所有分析工具的結果，為解決此問題，我們需在每一使用模組的 log 都加入一紅時間標記(time-stamped tag)。

有限記錄檔儲存空間：為客服記錄檔儲存空間不足的問題，我們一開始只分析一個模組

的效能，然後逐漸'限制性地'增加更多的分析模組和層次，以避免大量無關的結果。實做上，對於一個使用情景，例如，網頁瀏覽，我們首先選擇一些特定模組或其他專家所建議的模組來進行校能剖析。令 M 為被選模組集合，S 為其中被插入檢查點(checkpoint)的函式(function)，最後，這些含有 S 的 M 原始碼會被重建、執行並獲得分析結果。分析結果可包含不必要的 checkpoint，這些 checkpoint 必需予以刪除，以減少不必要的輸出。如此不斷重新生成新的原始碼，並重覆再次執行，以得到細化分析的結果。上述的步驟描述了一個分析階段(圖 3.1a)，經由多次迭代，從步驟 6 至步驟 10 步，可大幅減少 log 檔所需空間。然而，所選出的 M 可能無法涵蓋一個完整的執行流程所需的所有模組，對於這樣的情況，位於相鄰層(layer)有關的模組 M^+和含 checkpoint 的 function S^+會被加入，然後開始新的上述階段，一直重覆該做法，直到沒有新的模組為止。以下以 Android 系統為例，詳述實作方法。

開機時間(booting time)：開機時間被定義為從按下 Android 裝置的電源開關到'啟動螢幕'繪製完成(SurfaceFlinger.cpp 中的 SurfaceFlinger::bootFinished()執行結束)的這段時間。其中，Wi-Fi 服務和電信服務在我測試環境中是處於預設的'未啟動'狀態。

開機流程

按下電源開關後，Initial Program Loader (IPL) 會檢查硬體元件並啟動 Second Program Loader (SPL)，接著 SPL 會依序將壓縮的 kernel image 從 Flash 載到記憶體、將 image 解壓縮，然後執行 kernel 裡的第一個函式──start_kernel。kernel 首先初始化各個資料結構(data structure)和任務(task)，並加載入驅動程式。接著，user space 的程式依序初始化並開始執行 ──service_manager、deamon programs、zygote 及 media_server。Zygote 繼續預載一些 Java class 和 resource，以加速 Java 應用程式，然後啟動 system_server。system_server 是一個管理所有的 Android 服務的程式，。它可以掃描 flash，找出所有已安裝的應用程式，然後為每一程式創建 thread 以提供服務。其中一個重要的服務是負責螢幕拖曳的 SurfaceFlinger，在啟動過程也結束於 SurfaceFlinger 裡的 bootFinished( ) 執行完成。

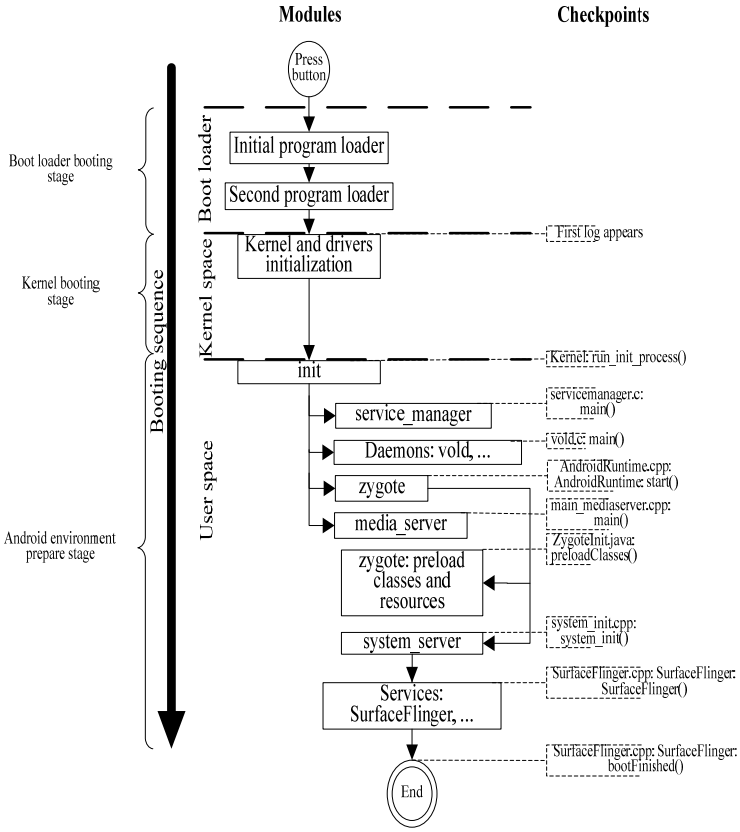


圖 1 開機流程

Checkpoint

根據上述啟動程序，我們分為開機過程分為三個區塊：導引程式、kernel space 和 user space，最初的三個 checkpoint 即被設置於這三個區塊的邊界。然後，我們使用了階段性插入的方法的逐啟動效能分析，並確定了以下七個重要 checkpoint：service_manger、daemon、zygote、media_server、JAVA class & resource、system_server 以及 SurfaceFlinger()。(圖 1)為上述過程。

網頁瀏覽：網頁瀏覽工作啟動時間被定義為從點擊螢幕上的瀏覽器後、在 Web 瀏覽器指定一個 URL，並完成載入指定的網頁(BrowserActivity.java 裡的 onPageFinished( )執行完成)的這段時間。

網頁瀏覽流程

點擊螢幕上的瀏覽器後會產生一個系統事件，該事件由 kernel 觸控螢幕的 driver 所產生的，然後被傳遞至 ActivityManager 和 Web 瀏覽器，Web 瀏覽器就可以知道指定的 URL 並把 request 發送到 Android 內建的瀏覽器引擎——WebKit。 Webkit 的使用 HTTP protocol 來請求所需的網頁，然後將收到的網頁解析到 Document Object Model (DOM) tree。通常，一個網頁包含多個原件（例如，圖片）當螢幕上準備畫出網頁時，Webkit 會繼續下載這些未取得的原件。

Checkpoint

在使用階段性插入效能分析時，模組 M 在第一次迭代中只含有 Web 瀏覽器，在經過多次反覆迭代後，我們可確定了以下重要 checkpoint：觸控式螢幕的中斷請求（IRQ）、Web 瀏覽器和 ActivityManager 的事件處理器、Webkit 的喚醒函式、HTTP 請求和回應函式、DOM tree 的建立和分析以及螢幕輸出功能。(圖 2)說明了這些重要的模組及其相對應的 checkpoint。

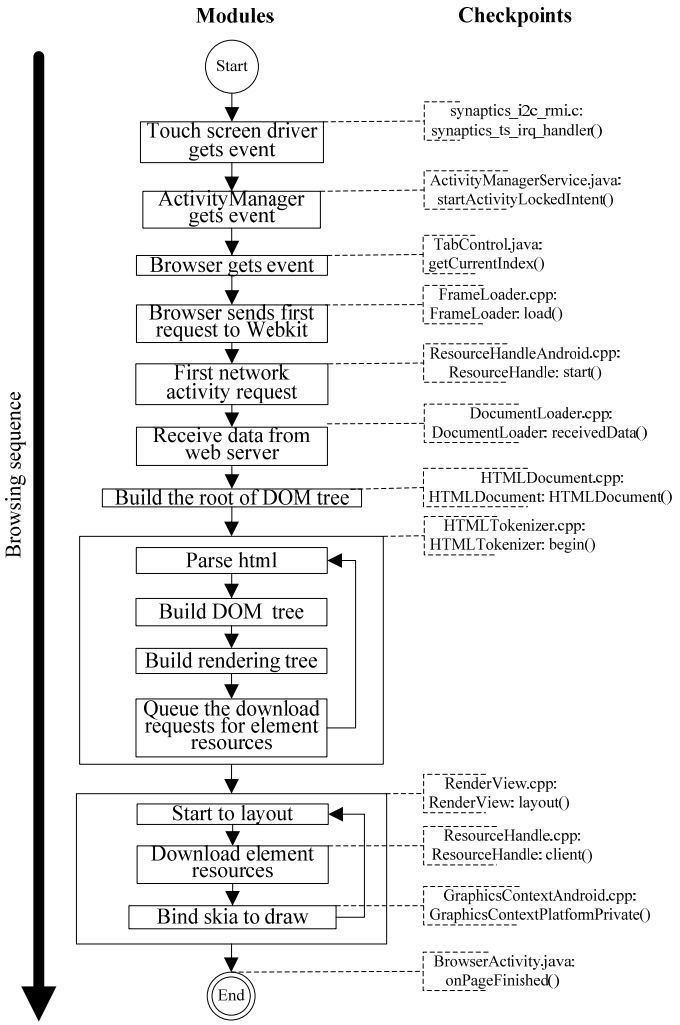


圖 2 網頁瀏覽流程

影音媒體播放：Android 內建的媒體播放器為 YouTube，所以在分析此情境之前，我們必需先連接到 Internet 並執行 YouTube，接著 YouTube 會把數個影片的圖示列在螢幕上。影音媒體播放時間被定義為從觸擊螢幕上的影片圖示到結束影片播放（即 MediaPlayer.java 中的 stayAwake()執行完成）的這段時間。

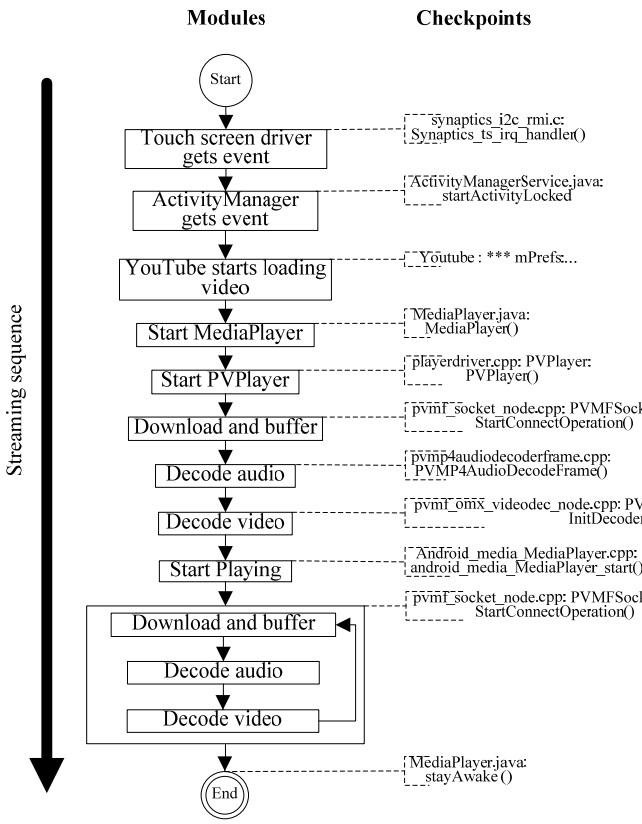


圖 3 影片播放流程

影片播放流程

如同網頁瀏覽，觸擊螢幕上的視訊圖示，也會產生一個系統事件，該事件會依序地傳送到 ActivityManager 和 YouTube，YouTube 接著會對 MediaPlayer 發出播放影片的請求，MediaPlayer 會再要求內建的影片播放器 PVPlayer 進行影片下載、解碼及顯示在螢幕上的工作。最後，整個播放流程結束於 stayAwake 函式。

Checkpoint

由於 YouTube 不開放其源始碼，因此我們無法直接對 YouTube 進行階段性插入效能分析，因此我們必需從 Android 的 application framework 層開始插入分析。在第一次迭代，M 包含了 application framework 層的模組，此外，以 binary 形式發佈的 YouTube 所留下的 log 檔也須加入 checkpoint。經過多次迭代，我們確定了以下重要 checkpoint：觸控式螢幕的中斷請求（IRQ）、事件處理程式 ActivityManager、記錄開始下載 YouTube 影片的 log 檔、MediaPlayer 和 PVPlayer 的建立、影音的下載及解碼、影片的播放以及最後的 stayAwake 函式。(圖 3) 說明了重要的模組流程

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 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將所使用的 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，不但有黑箱測試，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，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。

於 MLPP 中，我們也實際從現有 Android 裝置中測得了一些有效資訊。實驗結果顯示 72%的開機時間花在 user space 環境的初始

化，而在 user space 初始化中，44.4%的時間花在啟用背景服務程式及背景管理程式，37%花在先行載入 Java classes 和 resources. 實驗結果並顯示，網路是影響網頁瀏覽最主要的因素。在載入一個 2128 kB 大小的網頁的實驗環境下，螢幕繪圖顯示系統僅佔總執行時間的 5%。在 Wi-Fi 環境下播放一段 22 MB 的串流短片，系統需要花 10%的時間來做撥放準備的動作。影片資料下載加上音訊解碼部分的執行時間共佔了這段準備時間的 77%。從結果顯示，已能正確找出效能評頸並應用在實際例子上。

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.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

國科會補助計畫衍生研發成果推廣資料表

日期:2011/10/28

國科會補助計畫	計畫名稱: 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(中心分項)(2/2)
	計畫主持人: 林盈達
	計畫編號: 99-2220-E-009-044- 學門領域: 自由軟體暨嵌入式系統

無研發成果推廣資料

99 年度專題研究計畫研究成果彙整表

計畫主持人：林盈達	計畫編號：99-2220-E-009-044-
計畫名稱：嵌入式網路通訊裝置評比技術與工具之研發--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(中心分項)(2/2)	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	4	0	100%	件	
		權利金	2617	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	2	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	5	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	無

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：□已發表 ■未發表之文稿 □撰寫中 □無

專利：□已獲得 ■申請中 □無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

本子計畫目的在於提供一系列完整的嵌入式裝置效能測試工具與測試方法，為其建造「嵌入式網路通訊裝置測試軟體發展中心(Embedded Benchmarking Lab, EBL)」的一個分項。本子計畫的成果可用來協助廠商及開發者評比嵌入式網路通訊裝置的各項效能，進而對產品改良及修正，亦可做為日後學術相關研究時有實際的產品數據提供參考。於一般使用者更可依我們所提供的效能數據來參考及比較，選擇自己所需的產品。