以主從架構擴充使用 USB 埠連接 ADB 之數量
A Master/Slave Architecture for Extending the Number of ADB Connections Using USB ports

林敬文[1]、陳宥名[2]、陳偉凱[3]、周世邦[4]
財團法人資訊工業策進會[1, 2]
國立臺北科技大學資訊工程系[3, 4]
Ching-Wen Lin, Yu-Ming Chen, Woei-Kae Chen, Shih-Pang Chou
Institute for Information Industry[1, 2]
Department of Computer Science and Information Engineering,
National Taipei University of Technology[3, 4]
Email: {tedlin , randolphchen}@iii.org.tw,{wkchen, t103598019}@ntut.edu.tw

摘　要

Android 開發工具 Android Debug Bridge (ADB) 可以讓開發人員使用電腦操作 Android 裝置，例如模擬觸控、存取檔案等。但是，當需要操作大量的 Android 裝置時，受限於 ADB 的設計，同一部電腦最多只能以 USB 埠連接 16 台裝置。本論文提出一個主從架構，用來擴充以 USB 埠連接 ADB 之數量，稱為 USB Master/Slave 架構。Client 端首先連接 Master 主機，透過 Master 主機管控多部 Slave 主機，再由 Slave 主機各自連接大量的 Android 裝置，利用這種方式克服 ADB 連接上限的問題。此外，本論文以實驗比較三種不同的傳輸方式(USB Master/Slave、USB 、Wireless)執行 ADB 指令時的效能，實驗結果顯示使用 USB Master/Slave 時，只有輕微的額外消耗(overhead)，資料傳輸速率較使用 Wireless 更高，並且接近直接以 USB 埠連接裝置的傳輸速率。

關鍵詞：Android、ADB、USB Master/Slave。

一、前言

ADB (Android Debug Bridge) [1]是 Android 的開發工具，可以讓開發人員使用電腦操作 Android 裝置，例如模擬觸控、存取檔案等。但受限於 ADB 的設計，同一部電腦最多只能使用 USB 埠連接 16 台 Android 裝置，當有應用或服務需要同時操作大量的 Android 裝置時，就必須克服此限制。例如 Android 雲端測試服務平台[2][3][4][5]能提供 Android 應用程式開發人員進行 App 相容性測試的服務，此平台需要能同時操作大量的 Android 裝置(遠遠超過 16 台)，因此如何擴充 ADB 連接上限的數量，成為一個議題。

目前有三種克服 ADB 連接上限的方法。第一種方法是更改 ADB 原始碼，在阿里之家博客[6]提到 ADB 服務端掃描設備的端口號範圍為 5554~5585，相鄰的奇偶端口號組合起來控制一台裝置，因此 ADB 連接上限為 16 台裝置。因此，更改 ADB 原始碼，增加掃描的端口號，可以克服 ADB 連結上限的問題，但是連接的裝置可能會變得很不穩定。

第二種方法為使用 USB/IP 技術[7]，USB/IP 是一個使用 IP Network 來分享 USB 設備的專案，目的是讓 USB 設備的全部功能都能藉由網路分享於電腦之間。要分享的 USB 設備必須使用 USB 傳輸線連上作為 USB/IP server 的電腦，並在 server 上註冊要分享的裝置，如此 Client 端即可以透過 server 的 IP 位址取得 server 上分享的 USB 設備，但是，有些 Android 裝置無法與 USB/IP 相容，因此無法使用 USB/IP 控制該裝置。

第三種方法是將 USB 改用 Wireless[8]連接，直接透過 Wireless (WiFi)網路執行 ADB 命令，因此沒有端口號限制的問題，但是 Wireless 頻寬有限，會導致傳輸速率的下降。因此本論文提出替代上述三種方法的方案，以主從架構擴充 USB 埠連接 ADB 之數量。Client 端首先連接 Master 主機，透過 Master 主機管控多部 Slave 主機，再由 Slave 主機各自連接大量的 Android 裝置，利用這種方式克服 ADB 連接上限的問題。

本論文的組織如下，第二節介紹相關研究，第三節介紹 USB Master/Slave 設計與實作，第四節介紹各項連線測試之實驗設計方法以及結果分析，第五節為結論。

二、背景與相關研究

ADB 是用來與 Android 裝置溝通的命令列工具，藉由這個工具，我們可以直接操作與管理 Android 裝置，ADB 主要分為三個部分：

1. Server: 在裝置的背景中執行，為 Client 與 Daemon 間溝通的橋樑。
2. Daemon: 在裝置背景執行(稱為 adbd)，用來執行 server 轉發過來的 ADB command。
3. Client: 在裝置執行時，開發人員可發送 ADB command 至 server 端。

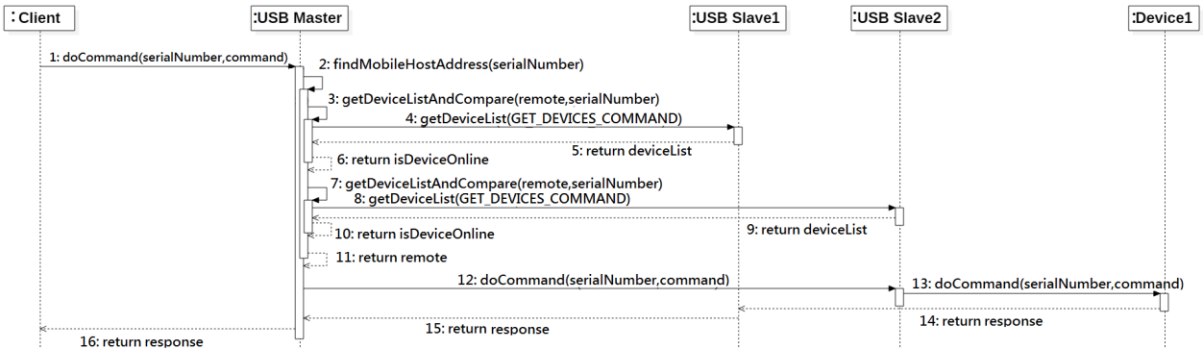

圖 1 USB Master/Slave doCommand 循序圖

指令通過 Client 交給 Server，Server 再和 Daemon 之間進行通訊。

另外 ADB 服務端(ADB fork-server server)掃描設備端口號範圍為 5554~5585，其中偶數端口號作為控制台連接，奇數端口號作為 ADB 連接，相鄰的兩個奇偶端口號組合起來控制單台裝置，因此單台電腦的 ADB 透過 USB 連接最多只能到 16 台裝置，透過更改 ADB 原始碼讓掃描的端口號範圍增加可以解決現問題，但是會讓電腦端連接 Android 裝置變得不穩定。因此，本論文並不採取此方案。

USB/IP 是一個使用 IP Network 來分享 USB 設備的 Open Source 專案，目的是讓 USB 設備的全部功能都能分享於電腦間。USB/IP 將"USB I/O message"封裝成 TCP/IP 的封包，藉此在 server 與 Client 端的電腦間傳遞封包。並且能在不改動任何原廠提供的驅動程式及應用程式的情況下，讓 Client 端的電腦使用遠端的 USB 設備，有如直接將遠端的 USB 設備安插於 Client 端電腦的主機上來使用。但是有些 Android 裝置無法與 USB/IP 相容。

ADB 命令除了透過 USB 執行以外，也可以透過 Wireless (WiFi)方式執行。也就是藉由 TCP/IP 協定連接客戶端與 Android 裝置，將原本透過 USB 來傳輸的命令改為透過 Wireless 執行，這種方法能夠克服單台電腦的 ADB 連接限制，但是無線網路頻寬的限制會影響 ADB 傳輸檔案的速率。上述三種方法各有缺點，因此，本論文提出 USB Master/Slave 架構作為解決方案。

三、USB Master/Slave 設計與實作

USB Master/Slave 是一個主從式架構，如圖 2 所示。Master 主機(以下簡稱 USB Master) 利用 Java RMI 的方式管控多台 Slave 主機(以下簡稱 USB Slave)，而每台 USB Slave 各自連接大量 Android 裝置，這樣的架構讓 Client 端的 ADB 指令能夠透過 USB Master 指派到適當的 USB Slave 執行，克服 ADB 連接上限的問題。舉例說，假設 Client 需要管控 48 台 Android 裝置，則可以設置 1 台 USB Master、3 台 USB Slave，各 USB Slave 分別管控 16 台 Android 裝置，則 Client 端即可透過 USB Master 管控 48 台裝置。由於 USB Slave 的數量可以任意擴充，因此，Android 裝置的數量亦可以任意擴充。

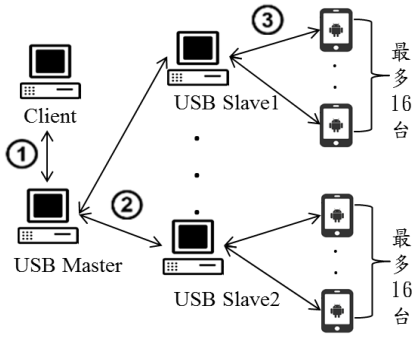

圖 2 USB Master/Slave 架構圖

USB Master/Slave 執行 ADB 指令的過程如下，Client 端對指定 Android 裝置執行 ADB 指令時，先將指令內容以及該裝置的 serialNumber 送往 USB Master，接著 USB Master 依序到每台 USB Slave 比對所有該 Slave 的 Android 裝置資訊，找到指定裝置的 Slave 主機之後將 ADB 指令送往該 USB Slave 執行，並將執行結果經由 USB Master 回傳到 Client 端，圖 1 為 USB Master/Slave 執行 doCommand 時的循序圖(doCommand 的功能詳見於表 1)。

3.1 USB Master、Slave、Device 之互動

USB Master、Slave、Device 之互動過程如以下步驟所示：

1. Client 對 USB Master 下指令: Client 端指定裝置(serialNumber)及對應的命令，要求 USB Master 執行。
2. USB Master 對 USB Slave 下指令: USB Master 依據裝置之 serialNumber，判斷裝置屬於哪一個 USB Slave，再委派該 USB Slave 執行指令。
3. USB Slave 對 Android 裝置下指令: USB Slave 要求 Android 裝置執行指令，並回傳結果。

表 1 USB Master/Slave Interface (最常用的五個命令)

Method	Description
connectDevice(java.lang.String serialNumber, java.lang.String uniqueID)	Client 綁定 Android 裝置，每個 Android 裝置僅能被一個 Client 綁定，Client 僅能操作(doCommand)一個已經被自己綁定的 Android 裝置，Master 透過 serialNumber 找到指定 Android 裝置的 Slave，下達綁定的指令，Slave 會將手機與 Client 的綁定資料登記，並回傳結果。
disconnectDevice(java.lang.String uniqueID)	將被 Client 綁定的 Android 裝置解除，Master 透過 serialNumber 找到指定 Android 裝置的 Slave，下達解除綁定的指令， Slave 會將 Android 裝置與 Client 解除綁定，並回傳結果。
doCommand(java.lang.String serialNumber, java.lang.String command, long timeout)	對指定的 Android 裝置(serialNumber)下 command ，Master 透過 serialNumber 找到指定 Android 裝置的 Slave，下達測試指令(ADB command)，Slave 再對 Android 裝置下 ADB command。
getFile(java.lang.String serialNumber, java.lang.String filePath)	取得 Slave 上的檔案，Master 透過 serialNumber 找到指定 Android 裝置的 Slave，下達回傳檔案的指令，Slave 會將檔案轉成 byte 格式回傳。
notifyMaster(java.lang.String slaveIPAddress)	通知 Master 有新的 Slave 加入或舊的重開，Slave 會 call Master 的 notifyMaster 來通知。

Client、USB Master、USB Slave 三者之間採用 Java RMI 方式連線。表 1 說明在 USB Master/Slave 架構下五個最常用到的 RMI Interface Method。

3.2 例外處理設計

USB Master/Slave 在執行過程中，可能出現例外狀況，其例外處理分為三種狀況:

1. 當 Client 與 USB Master 斷線時：USB Master 斷線時會拋出 RemoteException 通知 Client 端，再由 Client 端做適當的處理。
2. 當 USB Master 與 USB Slave 斷線時：USB Slave 斷線時會拋出 RemoteException 通知 USB Master，再由 USB Master 拋出到 Client 端，最後由 Client 端做適當的處理。
3. 當 USB Slave 與 Android 裝置斷線時：在每次下指令給 USB Slave 執行前，USB Master 會檢查該裝置是否連接成功，如果搜尋不到該裝置的話則回傳 null，並且印出找不到該裝置的訊息。

四、 效能評估

為了評估本論文所提出的 USB Master/Slave 架構的效能，我們以實驗的方式，比較在 3 種不同的傳輸方式(USB、USB Master/Slave 與 Wireless)下，執行 ADB 指令的效能。以下首先簡述實驗目的，接著介紹實驗環境以及設備，最後介紹各實驗之設計與結果分析。

4.1 實驗目的

理論上，當 Client 端直接以 USB 連接至某裝置時，執行 ADB 指令的效能應該是最好的(這種方法以下簡稱 USB)。而當使用 USB Master/Slave 架構時，Client 端所期望執行的 ADB 指令，必須先透過 USB Master 轉達指令至 USB Slave，再轉達至裝置執行，因此，執行指令時不免有些額外消耗(overhead)，我們希望評估這些額外消耗是否在可接受的範圍。另外，相較於 Client 使用 Wireless 方式執行 ADB 指令(詳見第二節)，USB Master/Slave 可以採用有線網路，傳輸效能理應較 Wireless 為高，傳輸品質也較穩定，因此，本論文設計 3 個實驗，驗證使用 USB Master/Slave 時，執行 ADB 指令的效率會比使用 Wireless 的效率來得高，傳輸品質更穩定，並且應該非常接近 Client 直接使用 USB 連接裝置之傳輸速率。

4.2 實驗環境

表 2 是實驗所使用的設備與環境。實驗時各主機(Client、USB Master、USB Slave)均使用獨立運行之電腦，以避免實驗數據受到外在因素所影響。

表 2 實驗設備與環境

電腦	
名稱	規格
Client	Windows 7 , CPU i7-4770@3.40GHz, RAM 8GB, 磁碟機 SSD W/R 540/480 MB/s
USB Master/Slave1	Windows 7 , CPU i7-4790@3.60GHz, RAM 16GB, 磁碟機 SSD W/R 540/480 MB/s
USB Slave2	Windows 7 , CPU i7-4790@3.60GHz, RAM 16GB, 磁碟機 SSD W/R 540/480 MB/s
網路線	
型號	Category 6 cable
Wireless AP	
型號	Buffalo WZR-1750DHP

當直接使用 USB 連線時，Client 主機直接以 USB 埠連接裝置，此時最大只能支援 16 台裝置，因此，實驗 3 省略部分 USB 之實驗。當使用 USB Master/Slave 時，Client 主機、USB Master、USB Slave 間均以 gigabit 有線網路連接，而各裝置則連接至 USB Slave 之 USB 埠。當使用 Wireless 時，Client 主機連接至無線 AP (Buffalo WZR-1750DHP)，再透過 WiFi 對各裝置下達 ADB 指令。

4.3 實驗設計與結果

實驗一：比較 8 種常用的 ADB 指令對單台 Android 裝置在 3 種不同傳輸方式下之效能

我們選擇最常用的 8 種 ADB 指令(詳見於表 3)做實驗。這些 ADB 指令中，除了傳輸檔案的 pull 指令，其餘指令回傳資料量都不大，因此不同傳輸方式應該對傳輸效能的影響不大。實驗設計如表 4。

表 3 實驗一之 ADB 指令(8 種常用的指令)

指令	指令功能
pull	將檔案從手機中傳輸到電腦端
ls	列出目錄下文件
pm list packages	找出待測程式、測試腳本與 Agent 是否已經安裝
am broadcast	以廣播方式啟動服務
input keyevent	啟用系統或軟體的 log 輸出
cp	複製手機上的檔案
devices	顯示在電腦上的手機
ps	顯示 Android 上所有正在執行的行程(process)

表 4 實驗一之實驗設計

實驗假設	除了 pull 指令傳輸效能有較大差異外，其餘指令差異不大。
實驗環境	1. USB:1 台 Client、1 台 Android 裝置。 2. USB Master/Slave: 1 台 Client、1 台 USB Master/Slave1、1 台 Android 裝置。 3. Wireless:1 台 Client、1 台 Wireless AP、1 台 Android 裝置。
實驗方法	Client 透過 3 種傳輸方式(USB、USB Master Slave 與 Wireless)測試 8 種常用的 ADB 指令(見表 3)，每個 ADB 指令各執行 10 次，紀錄平均執行時間，再比較各傳輸方式之差異。'pull'指令傳輸的檔案大小為 1MB。

實驗設備之連接方式如圖 3 所示，請參考第 4.2 小節的說明。本實驗僅使用 1 台 Android 裝置(HTC E8)，對該裝置重複執行 10 次 8 種 ADB 指令，實驗結果如圖 4 所示。由圖 4 可知常用的 8 種 ADB 指令中的 pull 指令傳輸效能的差異度較其他的指令大，顯示不同傳輸方式下執行傳輸檔案的 ADB 指令會有不同的效能，而其他指令則影響不大。因此實驗二和實驗三將針對'pull'指令進行進一步的實驗。

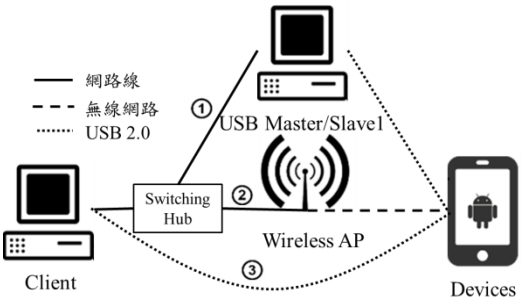

圖 3 實驗一、二之環境設備連接圖

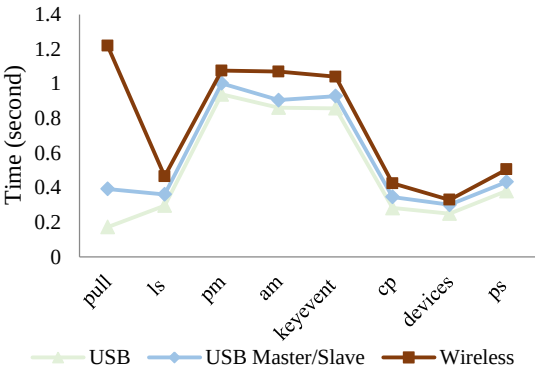

圖 4 八種 ADB 指令之傳輸時間

實驗二：在 3 種不同傳輸方式下，比較不同檔案大小對於 pull 指令的效能影響

本實驗由 Client 端對 Android 裝置執行 pull 指令，依序使用 3 種不同的檔案大小(100KB、1MB、10MB)，比較在 3 種不同的傳輸方式下，ADB pull 指令的效能差異。實驗中所使用的 Wireless AP 之傳輸速率理論值為 54 Mbps、USB 2.0 之傳輸速率理論值為 480Mbps，而 USB Master/Slave 同時使用 USB 2.0 與 Category 6 網路線(1Gbps)，其傳輸速率應低於 USB 2.0 但高於 Wireless。因此，我們預期 USB 應該最快，USB Master/Slave 其次，最後是 Wireless，實驗設計請參考表 5。

實驗對單台 Android 裝置重複進行 10 次 pull 指令，實驗結果如圖 5。由圖 5 可知 Wireless 的表現最差，在傳輸 10MB 檔案的情況下，USB Master/Slave 的傳輸速率為 Wireless 的 3.8 倍，因此實驗顯示使用 USB Master/Slave 當作傳輸方式，可以增加 ADB 指令的傳輸速率。相較於 USB，USB Master/Slave 則稍微慢一點，不過差異並不大。

表 5 實驗二之實驗設計

實驗假設	USB 最快、USB Master/Slave 其次，Wireless 最慢。
實驗環境	同表 3 之實驗環境。
實驗方法	透過 Client 將測試檔案(分別為 100KB、1MB、10MB)透過 ADB 傳送至單台 Android 裝置，紀錄平均執行時間，統計重複 10 次之實驗數據後，再比較各傳輸方式之差異。

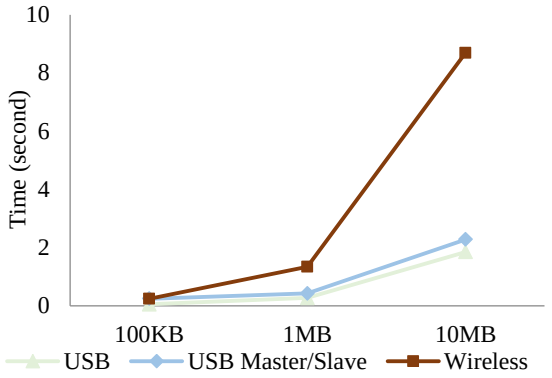

圖 5 ADB pull 指令之傳輸時間

實驗三：當 Client 端同時對多個 Android 裝置執行指令時，比較不同傳輸方式之效能差距。

本實驗由 Client 端同時對多個(分別為 8 台、16 台、24 台)Android 裝置執行 pull 指令，依序使用 3 種不同的檔案大小(100KB、1MB、10MB)，比較在 3 種不同的傳輸方式下，ADB pull 指令的效能差異。由實驗二結果可知USB傳輸速率最快，USB Master/Slave 其次，最後是 Wireless。因此，我們預期在多台 Android 裝置同時進行測試時，USB 傳輸速率受影響幅度最小、USB Master/Slave 其次，Wireless 最大，實驗設計請參考表 6。

表 6 實驗三之實驗設計

實驗假設	USB 受影響幅度最小、USB Master/Slave 其次，Wireless 最大。
實驗環境	1. USB:1 台 Client、16 台 Android 裝置。 2. USB Master/Slave: 1 台 Client、1 台 USB Master/Slave1、1 台 USB Slave2、24 台 Android 裝置。 3. Wireless:1 台 Client 端、1 台 Wireless AP、24 台 Android 裝置。
實驗方法	透過 Client 將測試檔案(分別為 100KB、1MB、10MB)透過 ADB 傳送至多台(8、16、24 台)Android 裝置，紀錄平均執行時間，統計重複 10 次之實驗數據後，再比較各傳輸方式之差異。

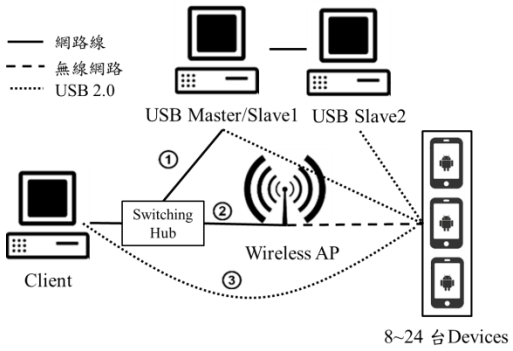

圖 6 實驗三之環境設備連接圖

實驗同時對多台 Android 裝置重複進行 10 次 pull 指令，實驗結果如圖 7、圖 8、圖 9。由圖 7、圖 8、圖 9 可知 Wireless 的傳輸速率隨著 Android 裝置的增多，受影響幅度最大，在傳輸 10MB 檔案的情況下，Android 裝置從 8 台增加到 24 台，Wireless 傳輸速率的成長幅度為 USB Master/Slave 的 3.6 倍。因此藉由此實驗顯示使用 USB Master/Slave 當作傳輸方式，傳輸品質比使用 Wireless 更穩定。在同時對多台 Android 裝置執行 ADB 指令的情況下，USB Master/Slave 與 USB 傳輸效能差異一樣不大。

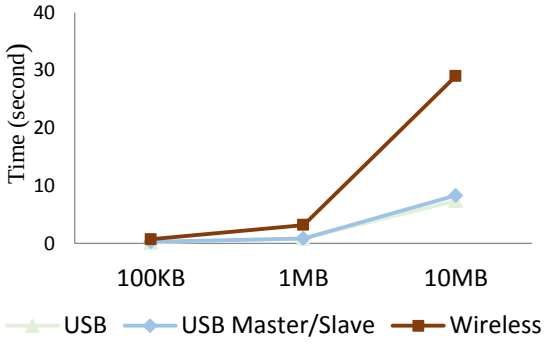

圖 7 ADB pull 指令對 8 台 Android 裝置之傳輸時間

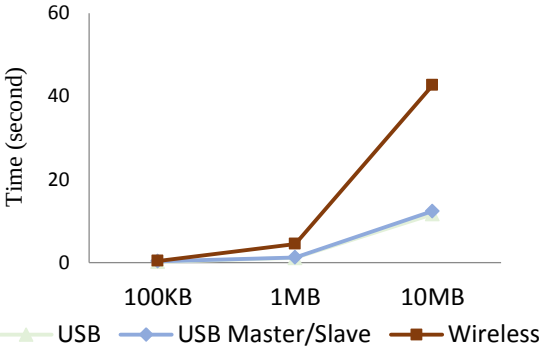

圖 8 ADB pull 指令對 16 台 Android 裝置之傳輸時間

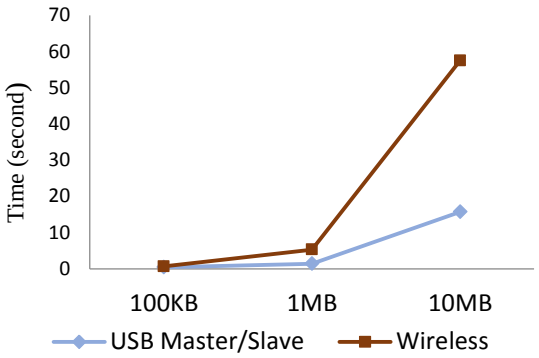

圖 9 ADB pull 指令對 24 台 Android 裝置之傳輸時間

五、 結論

本論文提出一個主從架構，稱為 USB Master/Slave，能解決電腦以 USB 埠連接 Android 裝置的數量上限的問題，實驗顯示使用 USB Master/Slave 連接 Android 裝置時，(1)資料傳輸速率方面確實比使用 Wireless 還高。(2)與 Client 直接使用 USB 連接裝置相比只有輕微的額外消耗(overhead)。(3)傳輸品質方面比 Wireless 更穩定。

致謝

本研究依經濟部補助財團法人資訊工業策進會「104 年度智慧手持裝置核心技術攻堅計畫(3/4)」辦理。

This study is conducted under the “The core technologies of smart handheld devices project ” of the Institute for Information Industrywhich is subsidized by the Ministry of Economic Affairs of the Republic of China.

參考文獻

[1] Android Debug Bridge, Available: http://developer.android.com/tools/help/adb.html
[2] 李友文，雲端測試服務平台 Android 測試過程錄影服務設計與實作，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[3] 賴勇安，一個以移除未使用的程式碼改善軟體維護性的方法：以 STF-CTP 為例，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[4] 劉淯鈞，Android 雲端測試平台的裝置連線品質改善方法，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[5] 朱宏文，Android App 伺服器效能測試平台，碩士論文，國立臺北科技大學資訊工程所，臺北，2015。
[6] 阿里之家博客, Available: http://www.blogbus.com/fafeng-logs/161910952.html
[7] USB/IP Project, Available: http://usbip.sourceforge.net
[8] Wireless ADB, Available: https://developer.android.com/studio/command-line/adb.html?hl=zh-tw#wireless