A Tree-Based Scatternet Construction Protocol for Improving Performance of Inter-Piconet Communication Service Over Bluetooth Radio Systems

林崇智*、張志勇**、許健平*
*中央大學資訊工程所，**真理大學資訊科學系
logistic@axp1.csie.ncu.edu.tw ; changcy@email.au.edu.tw ; sheujp@axp1.csie.ncu.edu.tw

Abstract

Bluetooth[10]是一種新的短距離無線通訊技術。裝有 Bluetooth 晶片的各個 device 可以在 2.4 GHz 的頻帶上以 FH(Frequency Hopping)的方式與其它裝有 Bluetooth 晶片的 device 建立一個 Piconet，並在 Master 的主控下彼此進行短距離的無線通訊，而多個 Piconets 可藉由 Relay 連結成一個 Scatternet，使不同 Piconet 的 device 可以透過 Relay device 的代傳訊息，進而彼此通訊。

由於每個 Piconet 是由 1 個 Master 及最多 7 個 active Slaves 組成，在形成 Piconet 時，服務的提供者及需求者很可能不處於同一 Piconet 中，為提供跨 Piconet 的服務，透過 Relay device 建立一條 routing path 將是有必要的。良好結構的 Scatternet 必須考慮到 routing 的效率、Master 的 linkage degree 及 Relay 扮演角色的個數[3]等特性。在這篇論文中，我們提出一 Clustering Protocol，使 Bluetooth system 能建立具良好通訊特性的 Scatternet。該 Protocol 乃以分散式的方法建構出一個 Tree-Based Scatternet，並提供了兩條 disjoint routing paths，而所建立 routing path 的長度將符合 O(log *n*)的最佳要求。

關鍵詞：Bluetooth、Clustering、Tree-Based

一、Introduction

近幾年來，無線通訊技術正以非常快的速度蓬勃發展，並與人類的日常生活息息相關。大家的生活普遍使用著手機、PDA、Notebook 等 mobile devices，利用 802.11 或 Irda 的無線技術來彼此通訊。然而，這些 devices 之間並沒有良好且一致的溝通介面，大都只有同一種類的 device 才可以彼此通訊。Bluetooth 則提供了一個良好的介面，可以使不同種類的 devices 透過它來彼此通訊。相較於 Bluetooth，短距離 802.11 或 Irda 的無線通訊技術，雖然可以使許多 devices 快速地彼此連結在一起，並提供 device 享有 internet 服務，也可用來做為取代 cable 的通訊媒介，但 Irda 的技術受限於 1 對 1、Line-of-Sight、30 度角的對齊、過短的傳輸距離及低穿透性的限制，所以僅使用於部分領域。而 802.11 的技術雖然提供高達 11Mbps 傳輸率、100 公尺左右傳輸距離及高抗干擾性，但受限於成本考量，所以僅作特殊用途的使用，並未大量使用於所有的行動裝置中。

Bluetooth 是一種具 10 公尺傳輸距離、低成本、FHSS 高抗干擾性與低耗能的無線通訊技術，其被操作在 2.4 GHz 的 ISM(Industrial Scientific Medical)頻帶上，而目前的傳輸速率為 1Msps。每個 device 皆有一個唯一的 48 bits BD_ADDR(Bluetooth Device Address)用來區分各個 device，並有助於 Master 產生一個特別唯一的 hopping sequence。多個裝有 Bluetooth 晶片的 devices 可以建構成一個 Piconet，而且各 Piconet 間也可以透過 Relay 做為彼此的通訊橋樑，進而形成一個 Scatternet。Scatternet 有助於擴大 Bluetooth 的通訊服務範圍，進而提供跨越其它 Piconet 所提供的通訊資源和服務[1][4][8]。

在一個 Piconet 中有 Master 和 Slave 兩種角色，一個 Master 在一 Piconet 中最多可以連接 7 個 active Slaves，且掌握了 Piconet 中運作的主控權。Master 只可以在某一個 Piconet 擔任 Master 角色，在其它 Piconet 中其角色必須為 Slave，而 Slave 可以同時參加多個 Piconets，並均扮演 Slave 的角色。服務於多個 Piconets 的 device 稱為 Relay，Piconets 之間可以透過 Relay 來做為訊息溝通的橋樑，Relay 藉由 TDM(Time Division Multiplex)的機制在多個 Piconets 之間切換以服務不同的 Piconets。而 Relay 可分為 Slave/Slave 及 Master/Slave 兩種。

Master 與 Slave 之間的通訊是利用 TDD(Time Division Duplex)方式，在 Master 的主控下，分別由 Master 與 Slave 輪替使用 channel，其使用 Time Division 的方式將 channel 切割成 slot 形式，每個 slot 的時間長度為 625μs。其 packet 的格式若以用途分類的話，則可分為 ACL、SCO、Control packet 三種；而若依 packet 長度來分類，則可區分為各佔 1、3、5 slots 等三種 size 的 packet。Master 可在偶數的 slot 開始傳送 packet，而則 Slave 在奇數的 slot 開始傳送 packet。

一個 Piconet 的形成主要必須經過 Inquiry / Inquiry Scan 及 Page / Page Scan 兩個重要的

procedures：

(1) Master 在起始時將進入 Inquiry State，依據其 Clock 值在 Inquiry Hopping Sequence 上的特定 channel 傳送 ID packet，其 packet 內容含有推衍自預設值之 Inquiry Access Code(IAC)。為了避免如果有兩個 Slaves 同時在同一個 frequency 收到 Master 所送出的 ID packet，而造成這兩個 Slaves 回傳 FHS packet 時發生碰撞情形，所以當每個 Slave 收到 Master 之 ID packet 後，必須先等一段 0~1023 個 slots 的 random Backoff 時間來避免這問題。

(2) 經過 Backoff 時間後，Slave 再重新進入 Inquiry Scan State 來接收 Master 傳來的第二個 ID packet。如果 Slave 正好處於 Inquiry Scan State 中並收到這個訊息，則會繼續接下來與 Master 通訊。

(3) 接收第二個 ID packet 之後，Slave 會將含有自己的 Clock 資訊及 BD_ADDR 的 FHS packet 回傳給 Master。Slave Clock 及 BD_ADDR 資訊，有助於 Master 在 Page State 時計算出 Slave 的 Paging Hopping Sequence。最後 Master 與 Slave 雙方分別進入 Page State 與 Page Scan State。

(4) 當 Master 進入 Page State 後，先傳送一個 ID packet 給 Slave，其 packet 內容含有推衍自 Slave 之 Device Access Code (DAC)，用來確認該 Slave 是否已經進入 Page Scan State。

(5) 若該 Slave 已經進入 Page Scan State，則回傳一個含自己的 DAC 的 ID packet 當 Ack。

(6) 當 Master 收到 Slave 所傳來的 ID packet 後，將再傳送一個 FHS packet 給 Slave，而此 packet 內容將包含 Master 的 Clock 及 BD_ADDR 資訊，使 Slave 能計算出 Master 連結後的 Channel Hopping Sequence。在此 FHS packet 中亦包含 3-bit 的 Active Member Address (AM_ADDR)以作為連結後 Slave 身份辨別之用。

(7) 當 Slave 收到 Master 所傳的 FHS packet 後，Slave 又回傳一個自己的 DAC 的 ID packet 給 Master，作為 FHS Packet 的 Ack。最後 Master 與 Slave 雙方皆進入 Connection State。

(8) Master 進入 Connection State 後，先傳送一個 POLL packet 給 Slave 做為 Piconet 連結成功確認訊息。

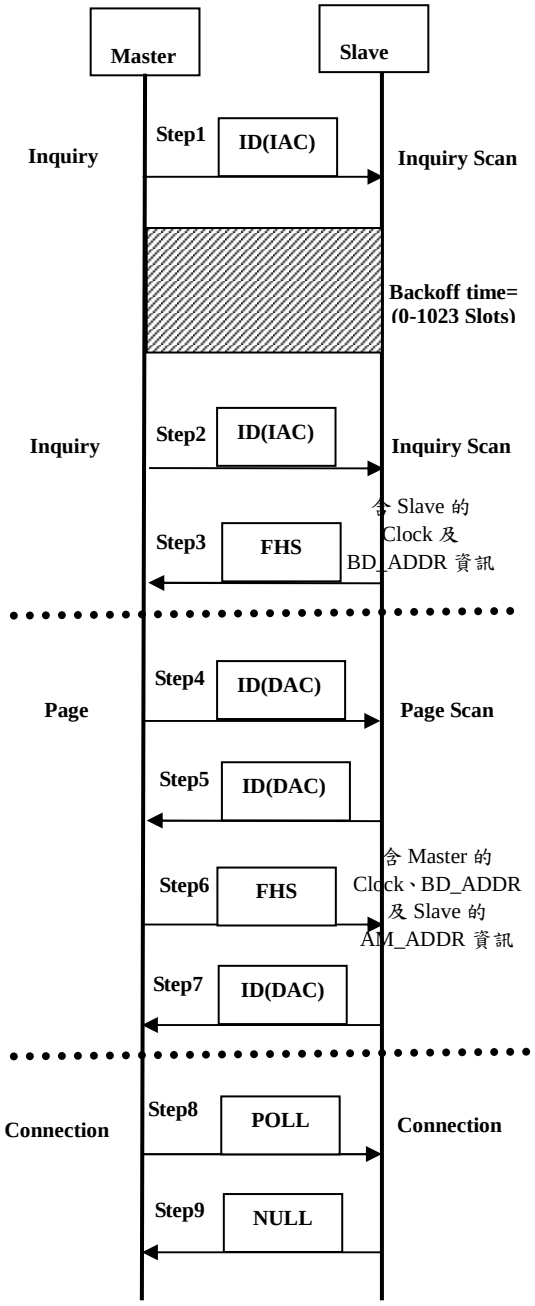
(9) 當 Slave 收到 Master 所傳的 POLL packet 後，Slave 又回傳一個 NULL packet 來告訴 Master 自己已經正式地加入 Master 所建立的 Piconet 了。

以下，我們將上述的 Master 及 Slave 其分別所處的 Inquiry / Inquiry Scan 與 Page / Page Scan 的主要流程以圖(一)及表(一)來表示。

由上述可知，Inquiry / Inquiry Scan procedure 發生在未知彼此資料的條件下雙方執行相互的溝通，由於 Master 及 Slave 必須分別處於 Inquiry 和 Inquiry Scan State 一段很長的時間，因此佔用了大部分的連結時間[6]，所以縮短花費在 Inquiry / Inquiry Scan procedure 的時間，將有助於加速 Piconet 的形成。此外，如果欲執行通訊的兩個 devices 不處於同一 Piconet，且彼此間存在有兩條 disjoint routing paths 的話，則可以避免在 routing path 上，突然有一個 intermediated device 發生 failure 而造成了無法傳送 data 的情況產生[9]及提供分散及加倍流量的功能。除了 Piconet 形成時間的縮短及 disjoint routing path 可影響 Scatternet 中任兩 devices 的通訊效率外，在一個 Scatternet 中，Piconet 個數少、Relay 及 Master 的 linkage degree 少也是建構一個好的 Scatternet 的必要條件[3]。Piconet 數量少將具有容易維護管理、減少 interference 及 balance 等性質，且 Relay 的 linkage degree 少可有效降低切換於不同 Piconets 間所需的 guard time overhead，而 Master 的 linkage degree 少將可避免執行 park / unpark procedure 的 overhead。因此，綜合上述，一個完美的 Scatternet 應具有下列的特色：(a) connected (b) Piconet 數量少 (c) Relay 的 linkage degree 少 (d) Master linkage degree 小於或等於 7 (e) 完美的 routing path 長度 (f) 任兩 devices 均具兩條 disjoint routing paths。

表(一)：Master 及 Slave 欲建立 Connection 時的詳細運作內容，包含 Message、Hopping Sequence、Access Code、Clock 及 State 的內容

Step		1	2	3	4	5	6	7	8	9
Message		ID	ID	FHS	ID	ID	FHS	ID	POLL	NULL
Hopping Sequence		Inquiry	Inquiry	Inquiry Response	Page	Page Response	Page	Page Response	Channel	Channel
Access Code and Clock		IAC Native	IAC Native	IAC Native	Slave's DAC	Slave's DAC	Slave's DAC	Slave's DAC	Master's CAC	Master's CAC
State of	Master	Inquiry	Inquiry	Inquiry	Page	Page	Master Response	Page	Connection	Connection
	Slave	Inquiry Scan	Inquiry Scan	Inquiry Scan Response	Page Scan	Slave Response	Page Scan	Slave Response	Connection	Connection



圖(一)：Master 及 Slave 欲建立 Connection 時所執行的 procedure

如何加速 Scatternet 的建立並使其具有 connected 及兩條 disjoint routing paths 的特性將是重要的研究議題。本論文提出一 Clustering Protocol，其以分散式的方式所建構出的 Scatternet 具有下列特性：較少的 Piconet 數量、Relay 的 linkage degree 少、同時存在兩條 disjoint routing paths、完美的 routing path 長度、不利用 park / unpark procedure 及 device 總數量不受限制等。

本 paper 章節安排如下：我們將在第二節中提出一些前人所研發的 Clustering 方法。在第三節中，則會詳細地描述我們所提的 Clustering Protocol 及建構 Tree-Based Scatternet 的 Protocol。在第四節中，我們以 Tree-Based Scatternet 為環境，述說 routing 的運作。在第五節中，我們將以實際數據來驗証其效率，並與前人所研發的方法作比較。最後在第六節中，做一個簡短的結論。

二、Background and Previous Works

一個好的 Scatternet Topology 可以使 Bluetooth 系統方便管理、容易維護及具較高的穩固性。在[5][6][7]中，分別提出 Clustering Topology 的 Protocol，使 Scatternet 能建構成 Mesh[6]或 Star[7]的 Topology，並分別符合 Piconet 數量最少、connected、完美的 routing path 長度及具兩條 disjoint routing paths 等特色，然而其 Protocol 執行時，皆需以 centralized 的方式選出一個 coordinator，來收集所有在範圍內 device 的 BD_ADDR 和 Clock 資訊，之後再執行 Master、Slave 及 Relay 角色的分配工作，進而完成一個 Scatternet 的建立。藉由 coordinator 來執行資料收集及角色分配的工作，會產生把大部分的 system load 皆加重於 coordinator，進而易於發生 system 不穩固及選取 coordinator 的 process time 過長等問題。在[6]中，為了建立一個 complete connected topology，更假設 device 個數 *n* 有上限，其中 *n* 值為 36。由於不斷地進行 Inquiry / Inquiry Scan Operation 的切換，以提高彼此對上的機率，所以很容易造成整個 Scatternet 形成時所需的時間花費是一個不易控制的變數。此外，[7][9]所發表的 Protocol 中，不容易找到兩條 disjoint routing paths，但同時存在兩條 disjoint routing paths 卻是非常重要的，處於不同 Piconet 的兩 devices 其單一通訊路徑將因某個 intermediated device 的消失、故障或是離開而造成了斷路，進而使系統不能運作而停擺。

在以下的章節中，我們提出有效率的 Protocol 來建構一個 Tree-Based 的 Clustering Topology，在 Protocol 的執行中，並不需一個 coordinator 做所有資訊收集的動作，完全採取分散式的方式來建立 Connection，且在角色分配工作上也只需指定少數 device 來扮演 Relay 的角色，其它 device 皆在 Piconet 建構成時，都已知道自己的角色。我們使用 State Pattern 的觀念來加速 Master 收集自己 Slave 的資訊，而不花費額外時間來收集非自己 Piconet 成員的資訊，這將有助於加速每個 Piconet 的形成進而使 Scatternet 的形成更為快速。此外，我們所建立的 Tree-Based Topology 也提供了同時存在兩條 disjoint routing paths 的特性，並使任兩 devices 之 routing path 均具有完美的長度。

三、Tree-Based Clustering Protocol

在討論 Clustering Protocol 之前，將先說明本論文的一些預設條件和環境，如下所述：

(1) 所有的 device 皆在可以建立連結的範圍內，也就是說任兩個 devices 皆有可能聽取到彼此所傳送出來的訊號。
(2) 每個 device 其 BD_ADDR (Bluetooth Device Address)的 LAP (Low Address Part)皆是獨立的，無任何關聯性。
(3) 每個 device 的電源啟動時間皆是獨立的，無任何關聯性。
(4) 存在一個 function *F* 可以藉由 LAP 電源啟動時間及 Current State Pattern 的輸入，平均隨機地輸出一個 Next State Pattern。
(5) 每個 device 分別維護 NOS(Number Of Slave)及 NOP(Number Of Piconet)兩個變數。此外，Device 可透過 FHS Packet 中 Undefined 及 AM_ADDR 等兩欄位來交換 NOS 及 NOP 的資訊。
(6) 在 State Pattern 中，1 個 Time Interval 期間，具有足夠的時間(>2FS+RB，[6])來完成 Inquiry / Inquiry Scan procedure 的執行動作。

以下我們將把 Scatternet 形成的動作分三個 Phases 來描述，Phase 1 主要目的是建構 Piconet；Phase 2 則是以 Phase 1 的 Piconet 為基礎來建構一個 Tree-Based Scatternet；Phase 3 則是加入 Thread 的功用，使 Tree-Based Scatternet 中不在同一 Piconet 的任兩 mobile devices 間具有兩條 disjoint routing paths。

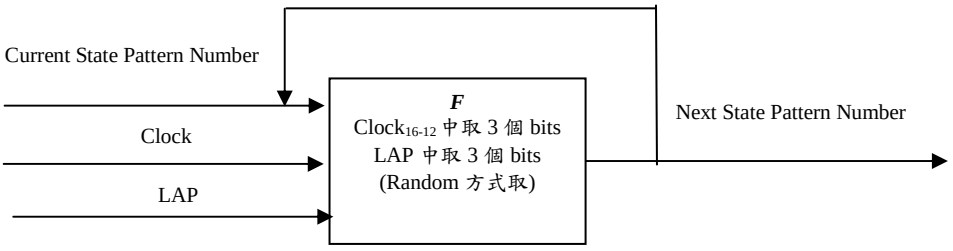
Phase 1：建構 Piconet

我們規劃了 7 組的 State Patterns，如表(二)所示。每個 device 藉由自己的 LAP、電源啟動時間及隨機初始的 Current State Pattern 編號為輸入，透用 function *F* 運算，決定出自己的 Next State Pattern。function *F* 的設計如圖(二)所示，每個 device 以其 $Clock_{16\text{-}12}$ 中的任 3 個 bits 的值及 LAP 中任 3 個 bits 的值與 Current State Pattern 的編號加總後 mod 7，來產生其 Next State Pattern 的編號，之後每經過 7 個 Time Intervals 會重新計算 Next State Pattern。每個 device 依照自己的 State Pattern 來扮演每個 Time Interval 時的角色，如表(二)所示。依照這樣的安排，在 Protocol 的初始時間可能有 1/7 的 devices 在執行 Inquiry operation，6/7 的 devices 在執行 Inquiry Scan operation。任 2 個 devices 就在隨機的況狀下，執行 Inquiry 與 Inquiry Scan operation 而有極高的機率彼此對上頻。雙方接下來就進行下面動作：每個 device 一開始的 NOS(Number Of Slave)值均為 0，先比較各自的 NOS 變數，而 NOS 的值是由處於 Inquiry Scan 者填在其 FHS Packet 的 Payload 中的 Undefined 及 AM_ADDR 欄位。當處於 Inquiry 者收到 NOS 值時，便與自己的 NOS 值比較，若 NOS 相等，則比較 BD_ADDR，比值較小的 device 會傳送之前獲得的 BD_ADDR 及 Clock 資訊和自己的 FHS packets 給 NOS 較大的 device，並且進入 Page Scan State 等待 Master Page 自己，由於 FHS packet 中含 BD_ADDR 及 Clock 資訊，Master 可以依此資訊精準算出 Paging Hopping Sequence，因此可順利與 Slave 建立連結。若 NOS 較大的 device 收集滿 6 個 FHS packets 則停止接收 FHS packet，而 NOS 較小的 device 則繼續依照自己 State Pattern 來執行收集 Slave 的動作。當 device 收集到 6 個 FHS packet 後，自動成為 Master，而直接去 Page 這 6 個 Slaves，且建構自己的 Piconet，並且進入 Phase 2。如此一來，大部分的 Master 皆可建構含 6 個 Slaves 的 Piconet，最多只有一個 Master 會建構未滿 6 個的 Slaves 的 Piconet。最後一個 Master 若在連續的 $7n$ ($n \geq 2$) 個 interval 皆沒有再對上其它 device，則直接去 Page 目前掌握的 Slave，建構自己的 Piconet。然後，進入 Phase 2。圖(三)、(四)、及(五)為 Phase 1 的範例，而表(三)為此範例中各 Steps 的詳細敘述。如圖(三)所示，圖(三)(a)中描述了兩個分別處於 Inquiry 及 Inquiry Scan State 的 device 其傳送 FHS 及 NOS 的過程，其中我們假設處於 Inquiry 者具有較大的 NOS；而圖(三)(b)中則另假設於 Inquiry Scan State 者具有較大的 NOS。圖(三)乃針對 2 個 devices 而言，描述其連結的細部動作，若環境中超過 2 個 devices 以上，則可能同時有許多對 devices 彼此互連，以下我們以環境中有 7 個 devices 為例來說明 phase 1 的流程。試考慮表(三)，若在 step 1 時 device 1 處於 Inquiry State 而其它 devices 處於 Inquiry Scan State，此時若 device 1 分別與 device 5 及 7 對上頻，則在比較過 NOS 後，若 device 1 為 Winner，則此時 device 1 已收到 device 5 及 device 7 的 FHS packets，如圖(四)所示，device 1 的 NOS 值將累計至 2，而此時 devices 1、5 及 7 將成為一 group，如圖(五)中的 step 2，我們以特殊的陰影來標示於這三個 devices。緊接著，在表(三)的 step 2 中，若 device 2 與 device 4 對上頻，device 2 在收到 device 4 的 FHS(含 NOS) packet 後，比較其 NOS 發現 device 4 為 Winner，device 2 將回傳一 FHS packet 給 device 4，因此，device 4 此時之 NOS 值為 1，如圖(五)中的 step 3，device 2 及 4 將組成另一 group(以垂直陰影表示)，爾後，如表(三)中的 step 4，當 device 1 與 device 4 對上

頻，由於 device 1 處於 Inquiry Scan State，因此當 device 4 收到 device 1 所送來的 FHS packet(含 device 1 之 NOS=3)後，比較得知 device 1 為 Winner，因此 device 4 回送一(含有 device NOS=1) FHS packet 給 device 1，因此，如圖(四)所述，device 1 的 NOS 將累積至 5，1 之 NOS 值累積到 6 時，便開始 Page 所收錄的 devices，最後 device 1 將形成一 Piconet 之 Master，並與 6 個 Slaves 連結而完成 phase 1，如圖(五)step 7 所示。

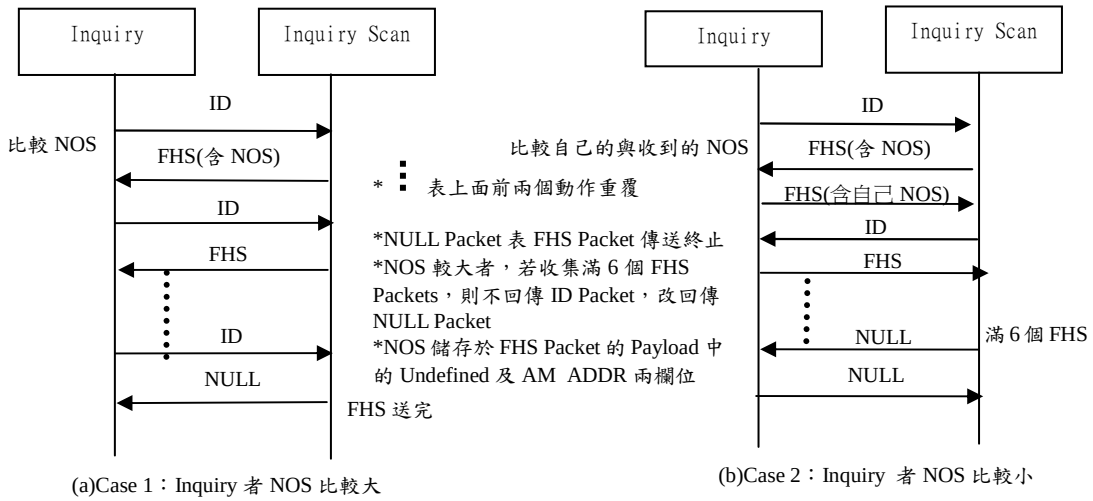
表(二)：以 Device 的 LAP、電源開啟時間及 Current State Pattern No 為 *F* 的輸入，輸出 Next State Pattern No，每 7 個 Time Interval 會再以 random 方式改變 State Pattern

State Pattern No	T_1	T_2	T_3	T_4	T_5	T_6	T_7	State Pattern No	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}
0	I	IS						2	IS		I	IS			
1	IS	I	IS					4	IS				I	IS	
2	IS		I	IS				6	IS						I
3	IS			I	IS			1	IS	I	IS				
4	IS				I	IS		3	IS			I	IS		
5	IS					I	IS	5	IS					I	IS
6	IS						I	0	I	IS					
I = Inquiry、IS = Inquiry Scan、T_N = Interval N , N = 1~7															



Next State Pattern Number = ***F*** (CLK, LAP, Current State Pattern Number) mod 7, Current/Next State Pattern Number ∈ {0,1,2,3,4,5,6}

圖(二)：Function ***F*** 的 finite state machine 的架構圖



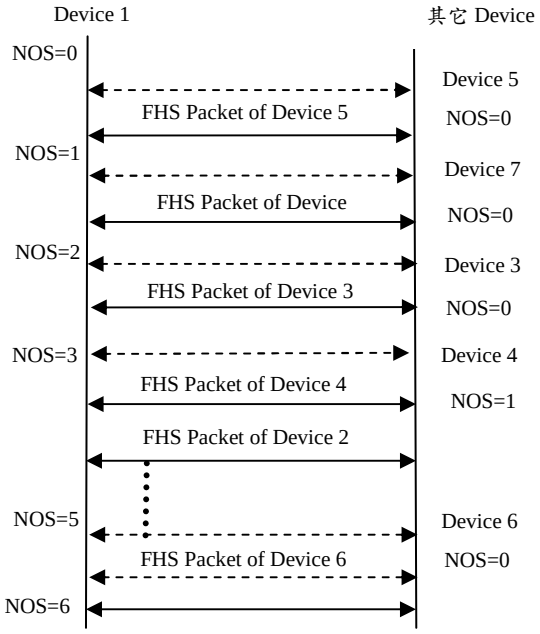
(a)Case 1：Inquiry 者 NOS 比較大

(b)Case 2：Inquiry 者 NOS 比較小

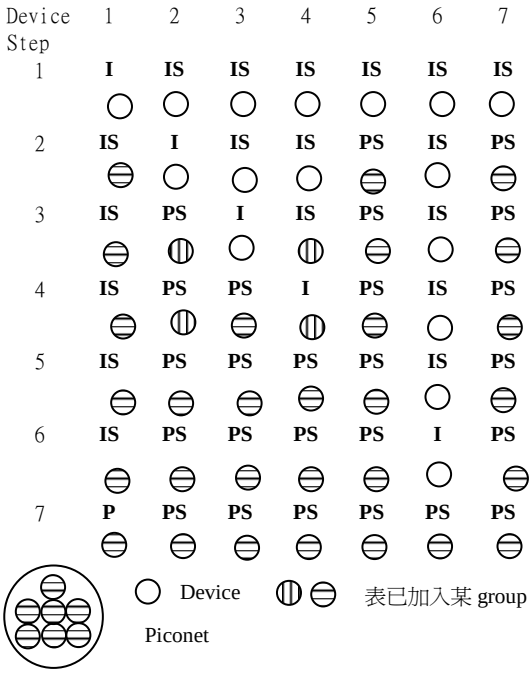
圖(三)：Device 在收集其它 Device 的 FHS Packet 時的詳細流程

依此類推，如表(三)中的 step 7 所述，當 device

◄- - -► 表對上頻



圖(四)：描述 Device 1 比較 NOS 及收集 FHS Packet 的流程



圖(五)：Piconet Construction Phase 的流程

在 Phase 1 中，已經建構完成大部份的 Piconet，為了可以提供跨 Piconet 的通訊服務，所以需要透過 Relay 來建構 connected Scatternet。經由 Phase 1 的執行，每個 Piconet 最多有 6 個 Slaves。1 個 Master 可連接 7 個 active Slaves，所以每個 Piconet 中每個 Master 至少還會有一個容量來容納 1 個 Slave，這個容量是被我們設計用來容納 Relay 的空間，以便連結其它的 Piconet，因此在 Phase 2 中，我們會利用這一個容量來執行 Master 對 Relay 的連結。

在 Phase 2 開始之時，每一個 Piconet 會從自己的 Slave 中選出 2 個 Slaves 來扮演 Relay1 及 Relay2 的角色嘗試與其它 Piconet 相連結，至於未滿的 Piconet 則由 Master 來當 Relay 的角色。對每個 Piconet 而言，由 Master 來執行 Inquiry operation，而被選定的兩 Relays 來執行 Inquiry Scan operation 分別與不同的 Piconet 的 Master 連結。剛開始時，一個 Piconet 可視為以 Master 為 Root 的 tree，如果不同 Piconet 的 Master 與 Relay 對上了，兩 Piconet 如何共同合併為一 Tree-Based Scatternet 如下所述：假設起始時每個 Master 記錄其 NOP 值為 1。

(1) 對上的雙方先比較彼此 NOP (Number Of Piconet)變數的值，若 NOP 值相等，則比較彼此 BD_ADDR。NOP 值填於 FHS packet 的 Payload 中的 Undefined 及 AM_ADDR 欄位。如圖(六)中所示，假設 Piconet A 中的 Master 與 Piconet B 中的 Relay 對上頻，Piconet B 中的 Relay 將傳送其 Master 之 NOP 值給 Piconet A 中的 Master，並由 Piconet A 中的 Master 比較彼此的 NOP 值。

(2) NOP 值比較大的 Piconet 其 Root，如圖(六)的 Case 1 中在 Piconet A 的 Master。尋找其所屬 Relay 中深度最淺的 Relay，將該 Relay 其 FHS packet 經由 NOP 比較大的 Master(或 Relay)傳送給 NOP 比較小的 Relay(或 Master)，之後再由該 NOP 值較小的 Piconet 其 Master (或 Relay)將 FHS packet 轉傳送給 Root，如圖(六)Case 1 中

表(三)：圖(五)中為建立 Piconet 時，各個 Device 在各 Step 中的 Action 及 Event

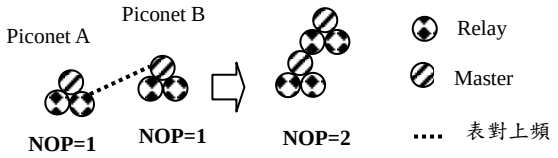
Phase 2：建構 Tree-Based Scatternet

Step\Device	1	2	3	4	5	6	7	Event
1	**I**	**IS**	**IS**	**IS**	**IS**	**IS**	**IS**	Device 5 和 7 與 Device 1 對上，Device 1 是 Winner
2	**IS**	**I**	**IS**	**IS**	**PS**	**IS**	**PS**	Device 2 與 Device 4 對上，Device 4 是 Winner
3	**IS**	**PS**	**I**	**IS**	**PS**	**IS**	**PS**	Device 3 與 Device 1 對上，Device 1 是 Winner
4	**IS**	**PS**	**PS**	**I**	**PS**	**IS**	**PS**	Device 1 與 Device 4 對上，Device 1 是 Winner
5	**IS**	**PS**	**PS**	**PS**	**PS**	**IS**	**PS**	No Event
6	**IS**	**PS**	**PS**	**PS**	**PS**	**I**	**PS**	Device 1 與 Device 6 對上，Device 1 是 Winner
7	**P**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	Device 1 Page Device 2,3,4,5,6,7，且建構 Piconet

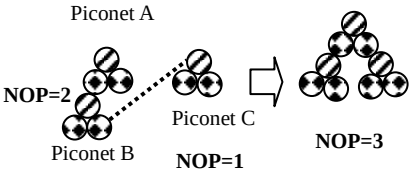
在 Piconet B 的 Master。最後由 Root 去 Page 該 Relay(以圖(六)的 Case 1 為例，即 Piconet A 的 Relay)，進而完成一個 Scatternet 的連結。

(3)持續地繼續進行下去，則某 Piconet 的 Master 將 具有很高的機率與另一 Piconet 的 Relay 建立 connection，而形成 Tree-Based 的 Scatternet，而兩個 Tree-Based 的 Scatternet 亦可執行同樣步驟，最後將會使所有 Piconet 連結成一個 Tree-Based Scatternet，由於每次均將某 Tree 的 Root 連結到另一棵 Tree 深度最淺的 Relay，因此 Tree 的高度將有效降低。若連續 n〔n≥ 2〕個 Interval 沒有與其它 device 對上頻的話，則直接進入 Phase 3。

最後，所有 Piconet 會連結成一個 tree topology 的 Scatternet。以上 Phase 2 的流程如圖(六)所示，同理，在圖(六)中的 Case 2 其處理情況亦為如此。在圖(七)中，我們將 Piconet A、B 及 C 的 Slaves 均省略，僅繪出 Master 及兩 Relays 與另一 Piconet 間的連結關係的示意圖。

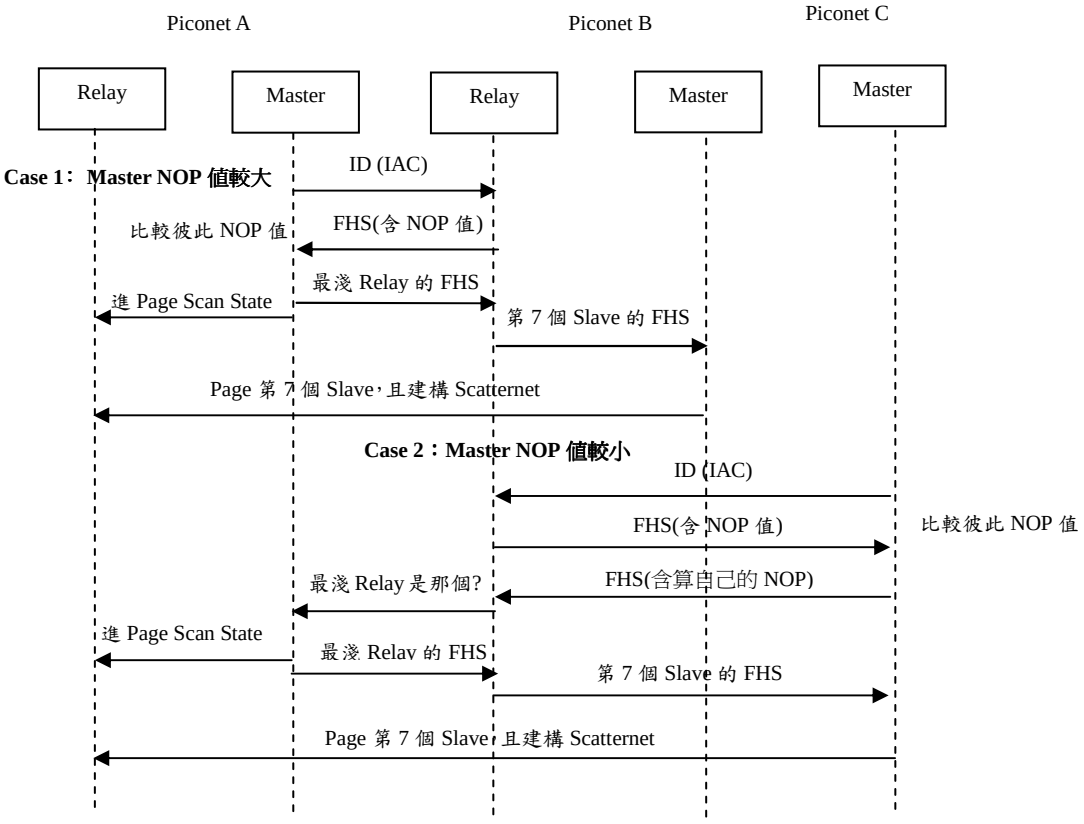


(a) 圖(六) Case 1 中兩 Piconets 的連結情形



(b) 圖(六) Case 2 中兩 Piconets 的連結情形

圖(七)：圖(六)中 Scatternet Construction Phase 中的示意圖。其中 Piconet 僅以 Master 及 2 個 Relays 表示，其餘 Slaves 省略。



圖(六)：當某個 Master 及 Relay 對上時，如何建構成一個新的 Scatternet 的詳細流程

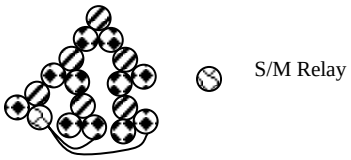
Phase 3：生成 Tree-Based Scatternet over

Master/Slave Relay

在 Scatternet 中，若提供了兩條 disjoint routing paths，則可以防止 routing path 中 intermediated device 不能運作時(可能因 power off 或移動性的因素)，所造成不能提供服務給這條 routing path 上的 device 的問題。為了使建立的 tree topology 提供兩條 disjoint routing paths，我們引用了 Threaded Tree [2]的特性，使 tree 中的 Leaf Piconet 彼此之間透過 Relay 建立另一 Piconet，其中少數的 Relays 必須扮演 Master/Slave Relay 的角色。如此一來，透過 Relays 建構而成 Piconets，將可改進原 Tree-Based Scatternet 不具備兩條 disjoint routing paths 的缺點，最後形成的 Scatternet 將具備有兩條 disjoint routing path 的特性。

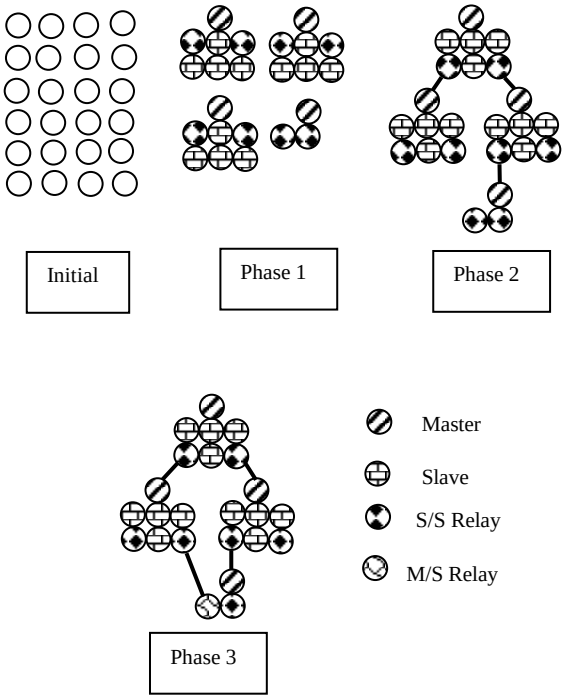
由 Phase 2 可知，在 Tree-Based Scatternet 中，最後會有個 Master 擔任 Scatternet 的 Root，該 Root 將負責尋找適當的 Leaf Piconet 的 Relays 來建構以 Master/Slaver 身份的 Relay 為基礎的 Piconet，以幫助 Scatternet 解決無兩條 disjoint routing paths 的問題。其進行的動作如下，(1)由 Root 去通知 Leaf Piconet 中的 Relays 在新建構的 Piconet 中各自要扮演的角色，並傳送包含 Slaves 資訊的 FHS packet 給所屬 Piconet 的 Master。(2)收到 Root 的通知之後，扮演 Master 與 Slave 角色的 device 分別進入 Page 及 Page Scan State 等待。(3)Master 對所屬的 Slaves 執行 Page operation，建立 connection。如此一來，我們所要建構的 Tree-Based Scatternet 就完成了。

Threaded Tree 的特性亦改進了在分散式環境下所建構出的 tree topology 的 Scatternet 其 routing path 長度不佳的缺點。因為在分散式環境下，每個 Leaf Piconet 的深度不一，所以可藉由 Root 來選擇深度較深與較淺的 Relay 連結成 Piconet，加上每次任兩個 Scatternet 的連結皆是透過深度最淺的 Relay 來連結，如此一來，則可以改善在分散式環境下所建構的 Scatternet 的 routing path 長度不佳的缺點。以上 Phase 3 的流程如圖(八)所示。



圖(八)：執行 Tree-Based Scatternet over Master/Slave Relay Construction phase 後所形成的 Topology。兩條線代表 Phase 3 中所增加的 Tread，使 Tree-Based Scatternet 具有 two disjoint routing paths。

以下的圖(九)及 Algorithm 的 pseudo code 分別用來解釋及分析上面的 Phase 1、2 及 3。



圖(九)：Scatternet 形成的流程

綜合上述，我們所提出的 Tree-Based Scatternet Construction Protocol 整理如下：

Phase 1：Piconet Construction Phase

1.1、device 依照自己的 State Pattern 執行 Inquiry / Inquiry Scan operation。

1.2、若與其它 Bluetooth device 對上了，則比較 NOS，輸的 device 送 FHS packet(包含之前收集的)給贏的 device，其中 FHS packet 內含有 device 的 BD_ADDR 及 Clock 資訊。輸的 device 停止 Inquiry / Inquiry Scan operation，進 Page Scan State 等待 Master 的 Page。

1.3、如果贏的 device 收集了滿 6 個 FHS packets，則停止 Inquiry / Inquiry Scan operation。自動成為 Master 角色，而直接去 Page 這 6 個 Slaves，建構所屬的 Piconet。然後進入 Phase 2。否則重覆 steps 1.1 及 1.2。

1.4、device 若連續 $7n(n \geq 2)$個 interval 沒有對上任意 device 的話，則自動成為 Master 角色，而直接去 Page 目前擁有資訊的 Slave，建構自己的 Piconet。並進入 Phase 2。

Phase 2：Scatternet Construction Phase

2.1、Master 由其所管轄的 Slave 中選 2 個 Slaves 當 Relays，Master 執行 Inquiry operation，兩 Relays 執行 Inquiry Scan operation。

2.2、若 Master 與另一 Piconet 的 Relay 對上了，則比較 NOP，贏的 device 會透過 Root 去尋找其深度最淺的 Relay 及要求其進 Page Scan State，並要贏的 device 將該 Relay 的 FHS packet 送給輸的 device，並傳送去給自己的 Root。

2.3、輸的 Root 去 Page 該最淺的 Relay，完成 Scatternet 的連結。

2.4、重覆 steps 2.1、2.2 及 2.3

2.5、若連續 n〔n≥ 2〕個 interval 沒有對上的話，則直接進入 Phase 3。

Phase 3 ： Tree Based Scatternet over Master/Slave Relay Construction Phase

3.1、由 Root 通知 Leaf Piconet 的部份 Relay 在新建構 Piconet 中所扮演的角色及所需的資訊。扮演 Master 及 Slave 角色的 devices 分別進 Page 及 Page Scan State。

3.2、分配到 Master 角色的 Relay 執行 Page 其所屬 Piconet 中的 Slaves 的動作。

3.3、完成最後 Tree-Based Scatternet 的建構。

四、Routing Protocol for Bluetooth Radio System

由於欲通訊的兩個 Bluetooth device 未必恰好同處於一 Piconet 中，因此 Routing 在 Scatternet 中扮演著非常重要的角色，前節中我們所建構的 Tree-Based Scatternet 將使 routing 更為便利。在這一節中，我們將探討在 Tree-Based Scatternet 如何有效率地來執行 routing。

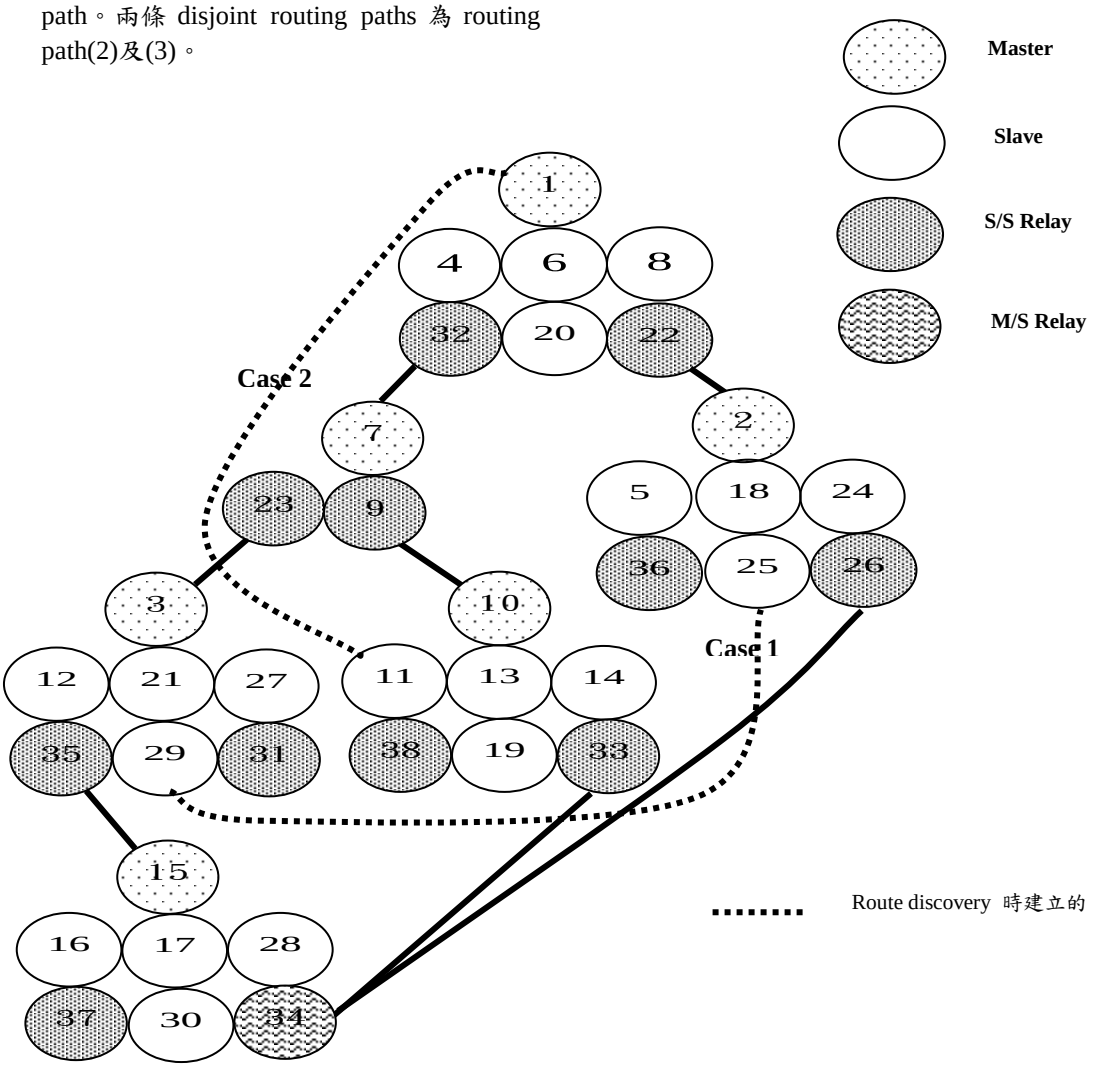
對於 routing 而言，我們必須就 destination device 的角色來區分成兩種情況，第一情況是 destination device 扮演為 Slave 的角色，另一種情況為非 Slave 的角色。首先，我們就 destination device 為 Slave 的角色的情況來探討，其流程如下所述。(1) source device 起始時會先以 flooding 的方式傳送一個有關於自己 BD_ADDR 及 Clock 的 packet 動作，並進入 Page Scan State。(2)當 destination device 收到 source device 傳來的 packet 後，因為所建構出的 Tree-Based Scatternet 具有 two disjoint routing paths(一條經過 Root Device，另一條不經過 Root Device)，所以會按原兩路徑先回傳 Ack packets。 (3) 再以所收到 Source Device 之 BD_ADDR 及 Clock 為依據來執行 Page source device 的動作，建立一個暫時的 Piconet。(4) 最後，以這個 One Hop 的 routing path 為主要的 routing path。所以在 source device 與 destination device 之間，扣除原先二條中較長的一條，則會存在有一條 one-hop 的及另一條以 flooding 方法所尋找到的等兩條 disjoint routing paths 用來傳送資料，且 routing path 的長度將符合 O(log *n*)的完美長度，其中 *n* 為 Device 個數。

另一種情況，當 destination device 為非 Slave 角色時，流程如下所述。(1) source device 起始時也是先以 flooding 的方式來傳送一個有關於自己 BD_ADDR 及 Clock 的 packet 動作，再進入 Page Scan State。由於我們所提出的 Tree-Based Scatternet 會必定存在有兩條 disjoint routing paths，一條由 source device 往上(Root 方向)搜尋，另一條由 source device 往下(兒子方向)搜尋。(2) 當 destination device 收到 source device 傳來的 packet 後，會按原兩路徑先回 Ack packets。(3) 若 destination device 少於 7 個 Slaves 時，則依第一種情況的方式處理，以建立 one-hop route。若 destination device 已有 7 個 Slaves 時，其中經過 Root device 的 routing path，在 Ack packet 回傳時，Root device 會執行 Page source device 的動作，與 source device 建立暫時的 Piconet，使 Slave 成為 Root device 的第 7 個 Slave。Root Piconet 因沒有 Parent Piconet，所以在 Phase 2 時其 Master 並不會與別的 Piconet 的 Relay 相連結，因此 Root Piconet 的 Master 最多只會有 6 個 Slaves。如此一來，source device 與 destination device 之間，則會有一條由 source device 連結 Root device，再由 Root device 到達 destination device 的 routing path，及另一條原先不經過 Root device 而 flooding 尋找到的 routing path 等兩條 disjoint routing paths，且 routing path 的長度亦符合 O(log n)的完美 Hops 數。以上兩種情況的流程如圖(十)所示。圖(十)為使用前節 Protocol 所建立好的 Tree-Based Scatternet，我們將以下例來解釋上述兩種情況的 Routing path 建立過程：

情況一：source 為編號 29 的 Bluetooth device，destination 為編號 25 的 Bluetooth device，由於 device 25 的身份為 Slave，依上述情況一的 Protocol 運作。device 29 將以 flooding 方式傳送自己的 BD_ADDR 及 Clock，並進入 Page Scan State。當 device 25 收到此 packet 後，先依據 packet 所經過的 path 回傳 Ack packet，之後進入 Page State 並和 device 29 建立一暫時的 Piconet，因此會存在一 one-hop 的 routing path。所以 device 29 與 device 25 之間會存在有三條 routing path：(1)

29 ⇔3 ⇔23 ⇔ 7⇔ 32 ⇔ 1 ⇔ 22 ⇔ 2 ⇔ 25；(2) 29 ⇔3 ⇔ 35 ⇔ 15 ⇔ 34 ⇔ 26 ⇔ 2 ⇔ 25；(3) 29 ⇔ 25；routing path(1)及(2)具有經過 shared intermediated device 與 link，所以選較短的 routing path 為第二條 routing path。兩條 disjoint routing paths 為 routing path(2)及(3)。

routing path(3)具有縮短 routing path(1)的功能，所以選擇它為兩條 disjoint routing paths 中的一條，致於另一條為 routing path(2)。



圖(十)：一具兩條 disjoint routing paths 的 Tree-Based Scatternet；以兩個 Cases 來解釋 Routing 的運作

情況二：source 為編號 11 的 Bluetooth device，destination 為編號 2 的 Bluetooth device，由於 device 2 的身份為已有 7 個 Slaves 的 Master，依上述情況二的 Protocol 運作。device 11 先以 flooding 方式傳送自己的 BD_ADDR 及 Clock，並進入 Page Scan State。當 device 2 收到此 packet 後，依據 packet 所經過的 path 回傳 Ack packet 給 device 11。在回傳 Ack Packet 途中，device 1 會在 Forward Ack Packet 後，先進入 Page State，之後與 device 11 建立一暫時的 Piconet。所以 device 11 與 device 2 之間會存在有三條 routing paths：(1) 11 ⇔ 10 ⇔ 9 ⇔ 7 ⇔ 32 ⇔ 1 ⇔ 22 ⇔ 2：(2) 11 ⇔ 10 ⇔ 33 ⇔ 34 ⇔ 26 ⇔2；(3) 11 ⇔ 1 ⇔ 22 ⇔ 2；

五、Comparisons with Related Works and Performance Study

以下我們針對許多相關 Topology 的論文與本論文所提的---Tree-Based Topology，做一些比較，並將其列於表(四)。由表(四)可知，為了容許大量 device 共同建構 Scatternet、避免 park 和 unpark 的動作、降低 guard time 的 overhead、提供 inter-Piconet 間任兩個 Bluetooth device 通訊所需的兩條 disjoint routing paths、並有較佳的 routing path 長度及不需具備 coordinator 的條件下，我們以 Tree-Based Topology 來形成 connected Scatternet，可由表(四)可得知，之前學者所提出的 Protocol 有下列的缺點---device 個數受限制、routing path 長

度太長、集中式的管理造成 Root device 成為整體 Scatternet performance 的 bottleneck，或有許多不確定因素，如執行 upark / park procedure 的 overhead 的問題，因此，採取這個 Tree-Based Topology 將有助於避免這些缺點，並保留之前學者所提出的 Protocol 的眾多優點。

Tree-Based Topology 所建構出來的 Piconet 及 Relay 的個數列於表(四)，由下面的算式可推得知表(四)的結果：

算式(1)：Phase 1 所建構的 Piconet 數量 $N_1 = \lceil D/7 \rceil$。

因 Phase 1 中每個 Piconet 大部分的 Master 皆有 6 個 Slaves，只會存在一個未滿 6 個 Slaves 的 Piconet。

算式(2)：Phase 3 所新建構的 Piconet 數量 $N_2 \leq \lceil 2^h/7 \rceil$，$h = \log_2 D$。

因 Phase 3 中會建立 Leaf Piconet 之間的連結，所以增加的 Piconet 數量是由 Leaf Piconet 的數量來計算。而在相同 node 個數下，complete binary tree 具有最多 leaf nodes，因此以 complete binary tree 的 leaf node 為上界。

由上面的算式(1)及(2)可得知，Tree-Based Scatternet 所建構出的 Piconet 的總算量 $N=N_1+N_2$，且 $N \leq (D+\log_x D)/7+2$

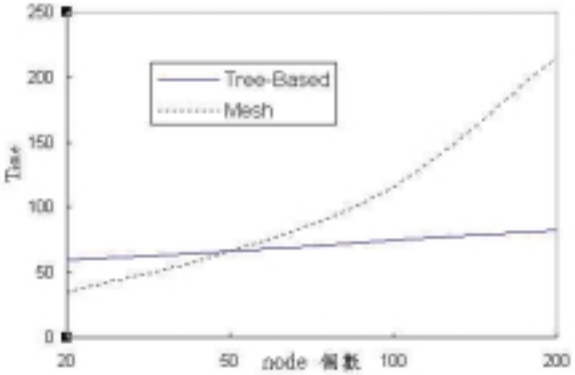
至於 routing path 長度及 Tree-Based Topology 能建構出存在兩條 disjoint routing paths，我們已在第四節中利用兩個範例來輔助說明。

圖(十一)、(十二)及(十三)分別為在不同的 device 個數狀況下，在 Inquiry 期間所花費的時間、Connection 期間傳送 Packet 的總數量及建構後的 Scatternet 包含的 Piconet 數量。我們的測試環境描述如下：在不同 device 數的情況下，我們隨機產生 device 的位置於一範圍內(任兩 devices 皆在通訊範圍)，device 的 BD_ADDR 及開機的 Clock 由隨機產生，我們針對連結時間、control packet 數量來觀察 Tree-Based Topology 的效率，並與 Mesh[6] Protocol 比較。分別在 device 點數為 20、50、100 及 200 的情況下做比較。其中 time 為一個相對基本單位。

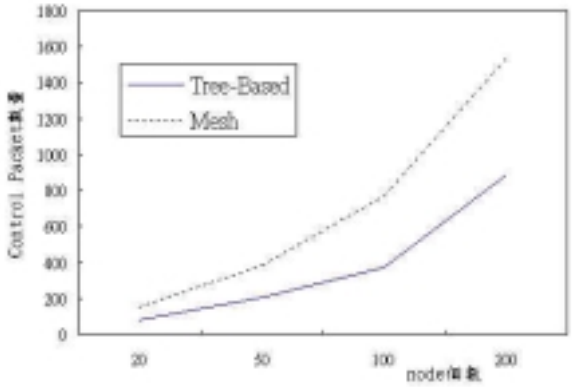
由圖(十一)可知，當 device 個數超過 50 個時，Tree-Based Protocol 會比 Mesh 的 Protocol 更節省花費在 Inquiry 的時間。device 個數少時，在 Mesh Protocol 的 Centralization 機制下，Coordinator 產生過程中比較次數較少，所以可快速地收集全部 device 的 BD_ADDR 及 Clock 資訊。

由圖(十二)可知，當 device 個數越多時，在 Connection 時，Tree-Based Protocol 的 Packet 總傳輸量比 Mesh Protocol 少越多。因 Centralization Algorithm 需要收集所有 device 的資訊，然後再分別送 Packet 給所有 devices 使他們知道自己該扮演的角色，如此一來，就會增加很多 Control Packets 的傳送。

由圖(十三)可知，Tree-Based Protocol 雖然在 Scatternet 中的 Piconet 總數量比 Ring Protocl (最佳值) 差，但其值已經很接近，所以 Tree-Based Protocol 是具備 Piconet 數量少的特性。



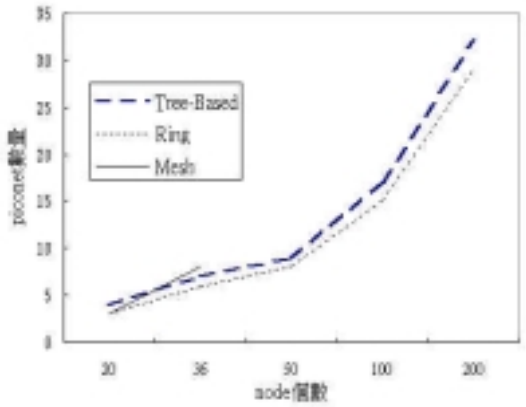
圖(十一)：執行 Inquiry 所需時間的比較



圖(十二)：建立 Connection 時所需的 Control Packet 數比較

表(四)：現有的一些 topology 與 Tree-Based 在一些條件的比較結果

Topology	two disjoint routing path	Routing path 長度	Device 總數量限制	Park Mode 需求	Piconet 個數	Relay 的個數	有 M/S Relay	分散式	Relay degree
Mesh[2]	Yes	O(1)	Yes	No	最多	C(D,2)	No	No	2
Star[5]	No	O(1)	No	Yes	1	0	No	No	0
Ring	Yes	O(N)	No	No	D/7	D/7	No	Yes	2
Tree-Based	Yes	O(log N)	No	No	$(D+\log_x D)/7+2$	$2*((D+\log_x D)/7+2$	很少	Yes	2
N = Piconet 的總數量(計算在下面討論)，D = device 的總數量									



圖(十三)：所建立的結構中，Piconet 數量的比較

六、Conclusion

這篇論文中，我們採用了 State Pattern 的方法，促使了我們可以快速地建構 Piconet；並利用 Master 和 Relay 分別進行 Inquiry 和 Inquiry Scan operation，來建構一個在分散式環境下的 Scatternet；最後藉由 threaded tree 的特性，讓 Leaf Piconet 的 Relay 建立以 Slave/Master Relay 為基礎的 Piconet 來解決 Tree-Based topology 的 Scatternet 不具備兩條 disjoint routing path 及 routing path 長度不佳的缺點，進而使我們的 Scatternet topology 保留之前學者所提出的 Protocol 的優點，並改善其造成的缺點。此外，由 performance 分析中，顯示出 Tree-Based Scatternet Topology 在整體比較上皆有不錯表現。在未來的研究工作上我們希望能超越 10 公尺通訊範圍的門檻。

References

1. Brent A. Miller and Chatschik Bisdikian, “Bluetooth revealed”, Prentice Hall PTR, 2001.

2. Ellis Horowitz, Sartaj Sahni and Susan Anderson-Freed, “Fundamentals of Data Structures in C ”, W.H.Freeman and Company, 1993.

3. Gergely V. Zaruba, Stefano Basagni, and Imrich Chlamtac,”Bluetrees-Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks”, IEEE ICC, pp273-277, 2001.

4. Jennifer Bray and Charles F. Sturman, “Bluetooth Connect Without Cables”, Prentice Hall PTR, 2001.

5. Lakshmi Ramachandran, Manika Kapoor, Abhinanda Sarkar, Alok aggarwal, “Clustering Algorithm for Wireless Ad Hoc Networks,” ACM, pp54-63, 2000.

6. LaMaire, ”Distributed Topology Construction of Bluetooth Personal Area Networks”, IEEE INFOCOM, pp1577-1586, 2001.

7. Manish Kalia, Sumit Garg, Rajeev Shorey, “Scatternet Structure and Inter-Piconet communication in the Bluetooth System”, IEEE National Conference on Communications 2000, New Delhi

8. Nathan J. Muller, “Bluetooth Demystified”, McGraw-Hill Companies Inc, 2001

9. Pravin Bhagwat, Adrian Segall, “A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets,” IEEE MoMuC, pp375-379, 1999.

10. The Bluetooth Specification, http://www.bluetooth.org