行動式 Ad Hoc 網路繞徑協定之實作與效能測試

Implementation and Performance Test of a Mobile Ad Hoc Network Routing Protocol

譚正中 曾煜棋 倪嗣堯 吳世琳 許健平

Cheng-Chung Tan Yu-Chee Tseng Sze-Yao Ni Shih-Lin Wu Jang-Ping Sheu

國立中央大學資訊工程研究所

Department of Computer Science and Information Engineering

National Central University

Chung-Li, Taiwan, ROC

摘要

在行動式無基地台無線網路（mobile ad hoc network）[1]中，一群 mobile host 各自擁有發送與接收訊息的裝置，各 mobile host 所送出的訊息無法經由基地台（base station）代為轉送，而必須由各個 mobile host 代為轉送。因為各個 mobile host 有可能隨意的移動，傳送封包的路徑也就隨之變動，所以各 mobile host 的繞徑表（routing table）之內容必須能因應網路拓樸變化而做改變，封包才能被正確地送達目的地。我們針對 ad hoc 網路環境的特性並參考目前已提出的幾種繞徑協定演算法，在 Linux 作業系統上進行實作；我們並在已實作完成的環境下進行測試，藉此觀察 ad hoc 網路的效能，並探討其中潛在的問題。

關鍵詞：行動計算、Ad Hoc 網路、繞徑協定、實作、效能測試

1. 介紹

大部分的無線網路環境皆使用到基地台，mobile host 傳出的資料都要靠基地台在傳統網路上幫忙轉送。但是行動式設備如筆記型電腦與 PDA 等等已漸漸被廣泛使用，他們可能在一些臨時的場合互相通訊（如戶外教學、地震與火山爆發的災難後現場），由於臨時佈線設置基地台很不方便，因此需要發展行動式 ad hoc 網路（mobile ad hoc network）。在此網路中，某一個 mobile host 送出的訊息要靠其他的 mobile host 代為轉送，但是每個 mobile host 都會移動，所以傳送資料的路徑必須及時建立，並且跟著 mobile host 移動的情形適時更改傳送的路徑，也就是每個 mobile host 的繞徑表必須能反應這種情況，才能建立或維護資料傳送的路徑。

一般在 wireless ad hoc network 環境中已提出的繞徑協定可分兩大類：proactive 與 reactive 繞徑協定。Proactive 繞徑協定意指每個 mobile host 間隔固定一段時間就會發送一些路徑相關資訊，各個 mobile host 就依據蒐集進來的資訊去改變自己的繞徑表，Distance-Vector protocol [8]就屬於這一類。Proactive 繞徑協定可以讓每個送出去的封包儘快得知到達目的地的路徑，比較不會有延遲；但是每個 mobile host 必須定期廣播訊息，所以相當浪費頻寬與 mobile host 的電源；若要降低廣播造成的大量頻寬的浪費，就要拉長每次廣播的間隔時間，這又可能會造成繞徑表不能及時反應網路拓撲的變化。

Reactive 繞徑協定只有在 mobile host 欲送出封包卻找不到路徑時才會運作。一些已提出的協定如 Dynamic Source Routing（DSR）[2,4,5]、Zone Routing Protocol（ZRP）[3]、Ad Hoc On Demand Distance Vector Routing（AODV）[9] 都屬於這一類。這類 protocol 最大好處是頻寬的使用量較小，只是某一 mobile host 欲送封包時，有可能需要重新尋找路徑，所以平均延遲時間較長。

就節省頻寬使用的角度來看，reactive 繞徑協定是比較好的協定，雖然封包傳送的平均延遲時間較長，但是在實際的環境中，延遲時間還不算太大。可是若能定期得知傳輸範圍中的其他 mobile host 的存在與否，對於繞徑表的維護也有很大的幫助，所以也不能否定 proactive protocol 的好處。所以我們設計的繞徑協定封包格式參考了 Dynamic Source Routing 協定[2,4,5]，繞徑的型態則延續現有的 next-hop 繞徑協定。

在 ad hoc 網路這個新的領域中，許多的問題必須經由實際測試才能得知其發生的現象、原因、及可能的解決方案，並瞭解實際環境的中一些限制，所以我們針對目前已經被提出的繞徑協定作了整理，並在 Linux 作業系統上進行實作，最後在實作出來的 wireless ad hoc 環境中進行幾項測試，如封包傳送的平均延遲時間、觀察廣播封包時可能發生的碰撞情況、與測量傳輸速率與品質等。

本篇文章第二節描述整個協定的巨觀行為，以便對

我們設計的協定有初步認識；第三節解釋我們所設計的繞徑協定之系統架構、協定中所使用的繞徑表結構與封包格式；第四節將仔細說明整個協定運作過程與程式設計的一些技巧與細節；第五節說明我們的測試的項目與效能的分析；第六節則是結論與提出未來可行的工作方向。

2. 繞徑協定的運作

我們設計的繞徑協定封包格式參考了 Dynamic Source Routing 協定，但資料傳送的方式則延續現有的 next-hop 方式遞送資料封包。

我們設計的繞徑協定重點有三個：第一，route discovery packet 之廣播與處理。第二，回覆與處理 route reply packet。第三，封包在傳送的途中發生問題之處理。

2.1. Route Discovery Packet 之廣播與處理

當某一 mobile host 欲傳送 packet 給某一個 destination host，packet 在建立 IP header 時便會檢查繞徑表，若找不到可到達 destination host 的路徑，此 mobile host 便會廣播 route discovery packet 以便尋找新路徑。收到 route discovery packet 的 mobile host 先檢查 packet 的 destination address 是否為自己，如果不是，先依據 packet 已有的 address sequence 資訊修改繞徑表，再填入自己的 IP address 到此 packet 中 address sequence 的尾端並將其再廣播出去，過程如圖 2.1 所示，因為 mobile host A 欲發送 packet 給 mobile host D 時找不到路徑，所以 mobile host A 便廣播 route discovery packet，mobile host B、C、D 與 E 會依據 route discovery packet 的內容修改繞徑表，並決定是否要繼續廣播。當 mobile host D 收到 route discovery packet 之後，一條由 host D 到 host A 的路徑便建立了，route reply packet 便可依此路徑反向從 host D 傳給 host A。注意將路徑記載到繞徑表或修改繞徑表中原有的路徑時，都會標上過期時間（expiration time）的欄位，也就是當此路徑過時（expired）時，此路徑就會由繞徑表中去除，以避免因路徑記憶太久不符合 mobile host 移動後的現狀。

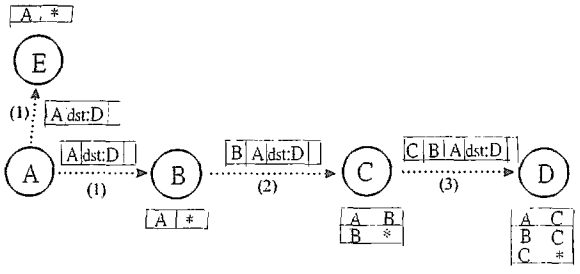


圖 2.1 route discovery packet 之廣播

為了避免產生路徑迴圈，每一個 route discovery packet 都配有一個 ID，當某一 mobile host 收到一個 route discovery packet 之後，先檢查之前是否有收過，若尚未收過，直接依據 route discovery packet 的 address sequence 更改繞徑表；反之，比較 route discovery packet 與繞徑表中到達 source host 的 hop count，如果較小就更新繞徑表，否則就丟棄。

2.2. Route Reply Packet 之回覆與處理

若 route discovery packet 所記載的 destination address 就是自己，先依據 route discovery packet 所記載的 address sequence 更改繞徑表，然後送出 route reply packet 給 source mobile host，途中的 mobile host 根據 route reply packet 中所記載的 address sequence 更改繞徑表，最後 source host 的繞徑表就含有到達 destination host 的路徑，一般的 packet 就可由 host A 送到 host D。如圖 2.2 所示。

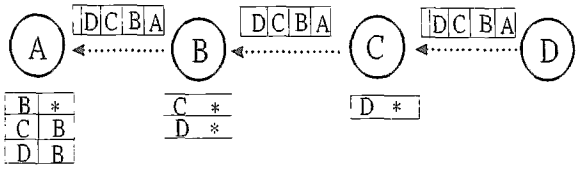


圖 2.2 route reply packet 之回覆與處理

2.3. Route Error 之確認與處理

當一個 packet 在傳送途中無法繼續被轉送下去，這可以分成兩種狀況來討論：

2.3.1. 負責轉送封包的 mobile host 不知道到達 destination hos t的路徑

在圖 2.3 中，host C 的 繞徑表不存在到達 E 的路徑，所以無法繼續轉送 packet，在大部份的情況下，mobile host E 應該不會跑太遠，這時 node C 便廣播一個小範圍的 route discovery packet，即限制 route discovery packet 所經過的 host 個數，即給定 TTL 一個上限值，圖 2.4 描述這個方法。假設 TTL 的上限值為 2，這個 Route Discovery 最多只會到達第二個 mobile host。TTL 的值可視為一個變動的參數，若網路結構變動很小，TTL 的值不必很大，反之 TTL 的值就必須加大。當網路連結變動非常劇烈時，此功能可以關閉。若發送了這個 route discovery packet 之後還是找不到一條可到達 mobile host E 的路徑，host C 就會送一個 route error packet 給 host A，當 host A 收到 route error packet 後就從頭廣播 route discovery packet 去找到達 host E 的路徑。

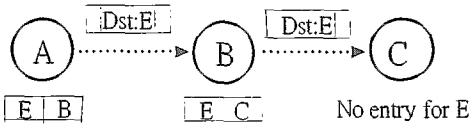


圖 2.3 封包在傳送途中找不到路徑

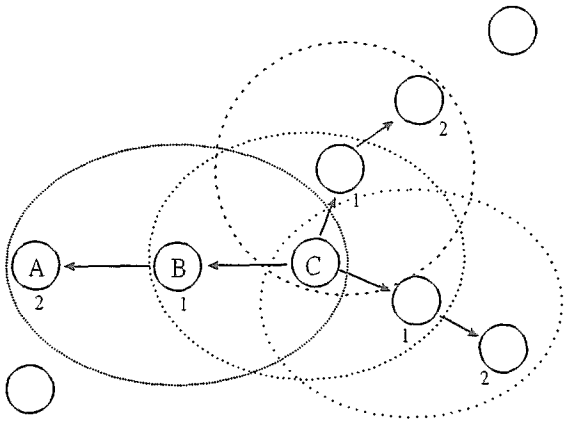


圖 2.4 廣播一個有限度的 route discovery packet

2.3.2. 繞徑表指示的 gateway 已不在通訊範圍內

某一 mobile host 的繞徑表中含有到達 destination host 的路徑，它會將 packet 傳送給繞徑表指定的 gateway host（即其通訊範圍內的某一個鄰居），這個 gateway host 收到 packet 之後會傳一個 Ack packet 給前一個 mobile host，我們的協定依據這個方法判定 packet 有沒有傳送成功。在圖 2.5 中，host C 沒有收到 host D 傳回的 Ack packet，所以就認定 host D 已經不在通訊範圍中。接下來 host D 處理此種 route erro r的方法就如同 2.3.1 節所描述的相同。

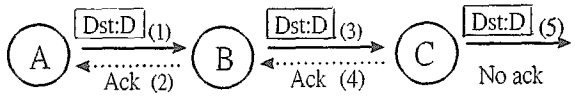


圖 2.5 另一種 route erro r情形

另外值得注意的是，在點對點（非廣播）封包傳遞過程中，為了確定接收端有正確的收到封包，通常在 MAC 層接收端會送一個 ack frame 回發送端。但由於 Linux 中的網路驅動程式界面並沒有提供是否有傳送成功的資訊，我們自行傳送 ack packet 以回應正確的傳輸，當沒有收到 ack packet 時，就知道先前的傳輸沒有成功。

3. 系統架構

3.1. 實作方式

因為我們所實作的協定是 On-demand protocol，必須能夠得知其他程式送出的 packet 是否可找到正確的路徑，這需要更改核心程式碼才有可能辦到，而 Linux 作業系統的原始程式碼是公開的，所以我們選擇在 Linux 上實作一個行動式 ad h o c網路動態繞徑協定。

目前 CMU 已經針對行動式 ad hoc 網路環境在 FreeBSD 上設計出 Dynamic Source Routing 協定[6]，他們將這個協定完全設計在 FreeBSD 核心程式中。而我們所設計的協定則分開在 daemon 與核心兩個部分，一些關鍵性的問題如 queue 的管理與繞徑失敗的察覺設計在 Linux 核心程式中，daemon 則負責繞徑協定中封包的傳送與繞徑表的維護。我們希望如此設計能得到兩個好處：第一，因為核心程式的設計非常繁雜與耗時，將關鍵功能放在核心碼中，其餘設計在 User Space 的 daemon 中，如此將可縮短協定開發的時程，我們便可以迅速開發各種不同的具實驗性質繞徑協定。第二，我們可提供新的系統呼叫，利用其設計出各種不同的 next-hop 這一類型的 wireless ad h o c繞徑協定

圖 3.1 顯示我們設計的系統在 Linux 作業系統中所處的位置，其功能區分為兩部份：(1)Protocol Daemon：負責維護繞徑表與發送 route discovery、route reply、route error packet。(2)Ad hoc layer：這是我們在 network stack 中新增加的一層，它管理因為找不到路徑而送不出去的封包、發送 Ack packet 與通知 Protocol Daemon 執行搜尋新路徑的動作。

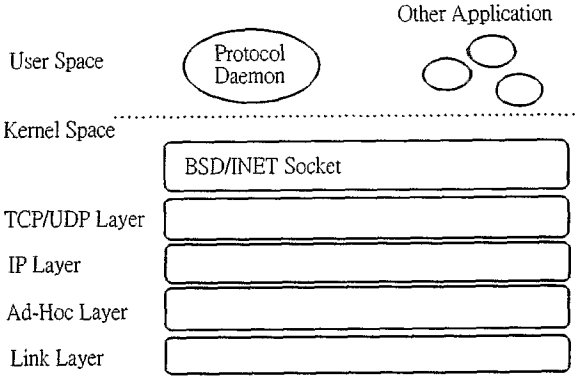


圖 3.1 系統架構

3.2. Ad hoc 繞徑表

在我們設計的協定中，我們不對目前網路架構中的繞徑模式做變動，送封包時仍查詢 IP 協定中原有的繞徑表，所以我們的繞徑方式是依舊屬於 next-hop 繞徑協定。但是為了能夠配合我們設計的繞徑協定而能動態維護原有的繞徑表，所以我們設計了一個 ad hoc 繞徑表，Protocol Daemon 透過 ad hoc 繞徑表中儲存的資訊才可以更改系統原有的繞徑表，而 ad hoc 繞徑表中每一個 ad hoc routing entry 的結構如圖 3.2。

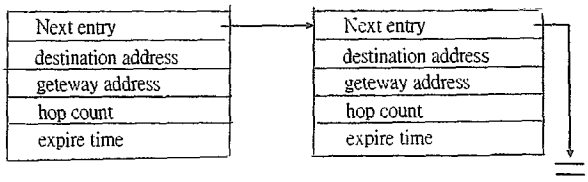


圖 3.2 Ad hoc 繞徑表

next entry：指向下一個 ad hoc routing entry。

gateway address：destination host 可能不會在通訊範圍內，所以必須將通訊範圍內某一個 mobile host

設定成 gateway，將封包送達 destination host 必須先靠這個設定為 gateway 的鄰居代為轉送。

hop count：將封包送達 destination 途中須經過的 mobile host 數目。

expire time：此 route entr y存活的到期時間。

3.3. 封包格式

Type	Len	HC	TTL	Identification	X
Target Address				Frist Hop(Source)	
Second Hop				Third Hop	
...				...	

圖 3.3 Route Discovery packet format

圖 3.3 顯示搜尋路徑時所用之封包格式。其中欄位說明如下：

Type : ROUTE_DISCOVERY。

Len : route discovery packet 的長度。

HC：即 Hop Count，記錄傳送途中已經經過幾個 mobile host。

TTL：即 Time To Live；限制所經過的 mobile host 個數。

Identification : route discovery packet 的識別碼。

X：備用欄位。

Target Address : Destination host 之位址。

First Hop、Second Hop、...：各個 mobile host 接收到封包必須將自己的位址依序填入的欄位。

Type	Len	HC	X	X
Last Hop(Target)				(N-1)th Hop
...				Frist Hop(Source)

圖 3.4 route reply packet format

圖 3.4 顯示路徑搜尋時回覆所用之封包格式。其中欄位說明如下：

Type : ROUTE_REPLY。

Len : route reply packet 的長度。

HC：即 Hop Count，記錄已經經過幾個 mobile host。

X：備用欄位。

Last Hop：原來 route discovery packet 中所指示的之 destination hos t之 address.

First Hop：原發送 route discovery packet 的 mobile host 之位址。

Type	Len	X	Target Address

圖 3.5 route error packet format

圖 3.5 顯示回報路徑錯誤所用之封包格式。其中欄位說明如下：

Type : ROUTE_ERROR。

Len : route error packe t的長度。

X：備用欄位。

Target Address：途中的 mobile host 欲轉送某一封包，卻找不到路徑，此封包的目的位址就被填入 route error packet 中的 Target Address field。

圖 3.6 顯示資料封包送達成功時之 acknowledge 封包形式。其中欄位說明如下：

Type	X	Target address
Neighbor address		X

圖 3.6 Ack packet format

Type : ADHOC_ACK。

X：備用欄位。

Target Address：原 packet 指定的目的位址。

Neighbor Address：發送 Ack packet 的 mobile host 之位址。

3.4. Kernel 中 ad hoc Send Queu e之結構

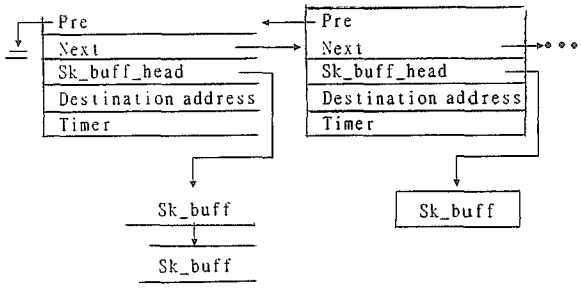


圖 3.7 ad hoc send queue entry

在 mobile host 中，當網路上層有 packet 要送，而路徑還不知道時，網路下層應先將此 packet 保留起來，並開始執行前述之繞徑協定；等到此路徑找到時，便可將此 packet 送出。此保留 packet 的機制就是 ad hoc Send Queue，由於是在 kernel 中運作，我們直接借用原有 Sk_buff 這個資料結構，並加上額外 Link head 指標，作為此 queue 的頭。以下說明此 queue 連結之欄位。

Pre：指向前一個 ad hoc queue entry。

Next：指向下一個 ad hoc queue entry。

Sk_buff_head：指到 Sk_buff 中的 destination address 與此 ad hoc queue ent ry的 destination 相同之第一個 Sk_buff。

Timer：在 kernel 中處理 ad hoc queue entry 時間到期，即此某一 entry 所管理的 Sk_buff 在一段時間後還沒被送出，就將此 entry 所有東西清除。

Sk_buff：在 Linux kernel 中管理每一個封包所使用的資料結構。

4. 系統設計細節

4.1. Route Discovery packet 之發送流程

整個流程如圖 4.1 所示。某一個應用程式送出一個 packet，檢查繞徑表時發現沒有到達 destination 的路徑，在 ad hoc laye r中就會將這個 packet 存放到 queue 裡面。接下來 kernel 送一個 signal(SIGUSR1) 給 Protocol Daemon，Protocol Daemon 透過系統呼叫 read_unresolved_information(...) 以得知它所要尋找的路徑為何，當取得足夠的資訊後，就利用 UDP 以廣播 IP 地址廣播一個 route discovery packet 去尋找新的路徑。

演算法：

User Space or Upper Network Layer:
Kernel or an application sends a packet;
Kernel Space (ad hoc Layer):
if (there is no route for the packet){
Record some information of the packet;
Queue the packet in the AdHoc send queue;
Send a signal(SIGUSR1) to Protocol Daemon;
}
Protocol Daemon:
The signal(SIGUSR1) is triggered by kernel:
Read the unresolved information;
Broadcast a Route Discovery packet;

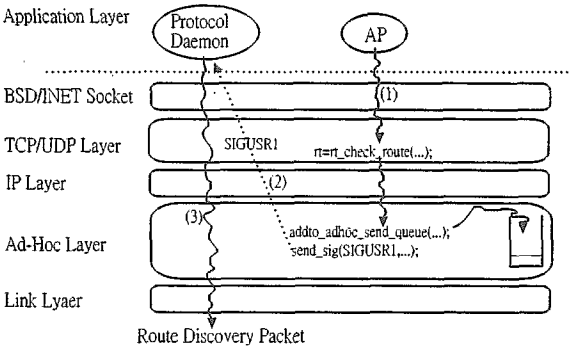


圖 4.1 啟動 Route Discovery

4.2. Route Reply Packet 之處理流程

某一 mobile host 收到一個 route reply packet 之後，查出此其 destination 為自己，然後經由系統呼叫更新繞徑表，透過系統呼叫 send_unresolved_packet(...) 將原本存放在 queue 裡面的 packet 送出去。

演算法：

Protocol Daemon:
Update routing table;
if(Source Address != my address){
Forward the Route Reply Packet;
} else {
Call the system call(send_unresolved_packet(...));
Kernel Space:
Send the packets in the AdHoc send queue that the destination address is matched;

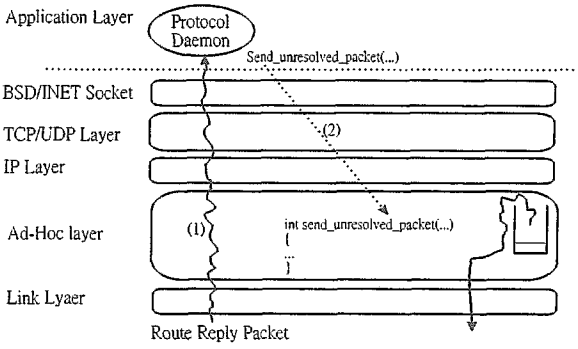


圖 4.2.Source mobile host 處理 Route Reply packet 之過程

4.3. Route Error 之處理流程

當一個 packet 在傳送的途中因故無法繼續傳送下去，即產生了 route error 的情形，這可以分三部分來說明：(1) 在圖 4.3 中，某一 mobile host 收進一個 packet，發現它的目的地不是自己，而繞徑表中並沒有存在到達 destination address 的路徑，這個 packet 就先被放進 queue 中，接下來 kernel 送一個 signal(SIGUSR1) 通知 Protocol Daemon 廣播一個小範圍的 route discovery packet 以找尋新的路徑。(2) 某一 mobile host 送出一 packet 之後經過一段時間還沒有收到 gateway host 傳回的 Ack packet，這個 gateway packet 可能已經不在通訊範圍內了，這時候也會送一個 signal(SIGUSR1) 給 Protocol Daemon 廣播一個小範圍的 route discovery packet 以找尋新的路徑。(3) 在經過一段時間之後 kernel 會送另一種 signal(SIGUSR2) 通知 Protocol Daemon 檢查是否已找到新的路徑，若沒有，就送一個 route error packe t給 source mobile host，通知它重新找一條到達 destination host 的路徑。

演算法：

Kernel (ad hoc layer):
if (there is no route for the packet && the source of the packet is not me) {
Record some information of the packet;
Queue the packet into the AdHoc send queue;
Send a signal(SIGUSR1) to Protocol Daemon;
}
Send a signal(SIGUSR2) to Protocol Daemon after a time interval;
Kernel (ad hoc layer):
if (the source address of the outcome packet is not me) {
Send an Ack packet to pre-host;
}
Record some information of an outcome packet that has a path to destination;
if (there is no Ack packet from gateway after the outcome packet is sent) {

Send a signal(SIGUSR1) to Protocol Daemon;

}

Send a signal(SIGUSR2) to Protocol Daemon after a time interval;

Protocol Daemon:

The signal(SIGUSR1) is triggered by kernel :

Broadcast a Route Discovery packet (limit TTL);

Protocol Daemon:

The signal(SIGUSR2) is triggered by kernel :

if(the path to destination is not find)

Send a route error packet to source;

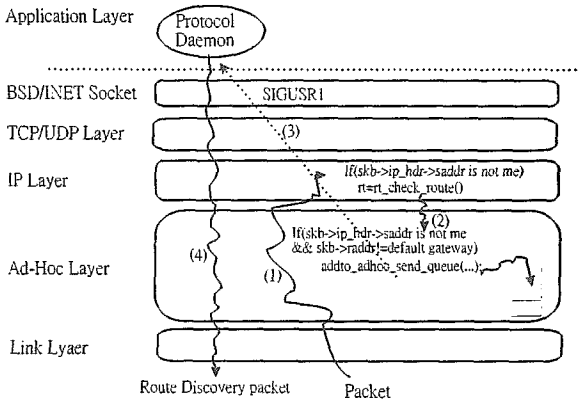


圖 4.3 處理 2.3.1 之 route erro r情況

5. 實際測試的環境與效能分析

我們使用三部筆記型電腦（Pentium 233MMX 一部、Pentium 200MMX 一部、Pentium II 350 一部）與一部 Pentium 200MMX PC，各配有一張 Lucent WaveLan IEEE PCMCIA 無線網路卡，採用 IEEE 802.11 MAC 協定，使用 2.4GHz 頻帶，傳輸速率最高可達 2Mb/sec，在一個良好無遮蔽物的室外環境中測試，其通訊半徑約 365m。

5.1. 傳送封包平均延遲時間

我們測試的方法如下：當 notebook A 欲送出一個封包給另外一部 notebook B，而 A 的繞徑表中並沒有到達 B 的路徑，我們計算尋找路徑所花費的時間，如圖 5.1 所示我們測試了三種狀況，在狀況 2 與 3 中 mobile host A 與 mobile host B 無法直接通訊，他們必須透過其他的 mobile host 來轉送封包，我們將每種情況各執行一百次得到每個送不出去的封包平均延遲時間，得到如表 5.1 的數據。

實際上測得的時間都非常短，從圖 5.1 的狀況 1 到狀況 3 分別花費的時間成倍數成長，而狀況 1 到狀況 3 的網路傳輸收送訊息的次數分別是 4、8 與 12 次。

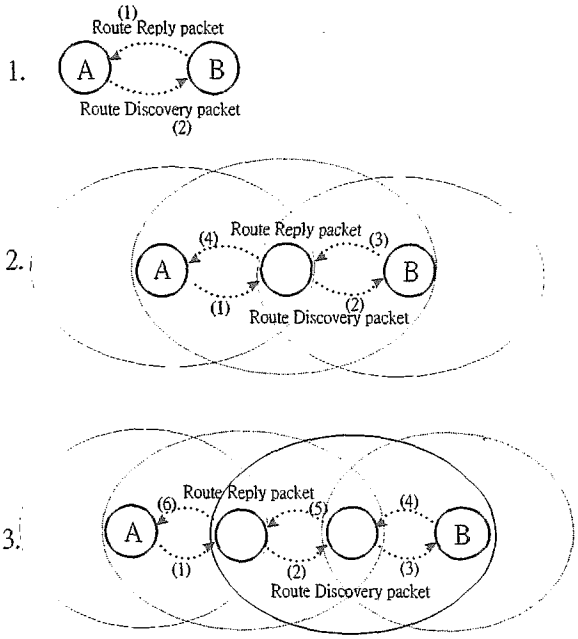


圖 5.1 測試平均延遲時間的環境

	Case 1.	Case 2.	Case 3.
Time(sec)	0.00385	0.00772	0.01076
Times(Tx&Rx)	4	8	12

表 5.1 封包傳送平均延遲時間

5.2. 同時廣播 Route Discovery packet 時的可能發生的碰撞情況

5.2.1. 碰撞發生原因

無線網路上之廣播[7]是一種不可靠的傳輸，某一 mobile host 送出一個廣播型封包之後不必等待回應，而 MAC 層靠著 carrier sense 得知 channel 無其他 mobile host 傳資料時，就將這個廣播型封包傳送出去，但是也只會傳送一次，不會有 backoff 數次的情況出現。這對於 ad hoc 繞徑協定中廣播 route discovery packet 就有可能出現碰撞，導致出現找不到路徑的情況。在圖 5.2 中，mobile host A 欲尋找可將封包送達 host D 的路徑，所以 host A 廣播一個 route discovery packet(傳輸(1))，host B 與 C 收到此一 route discovery packet 之後，都發現 channel 是空的，所以他們就各自(傳輸(2)、(3))將這個 route discovery packet 再次廣播出去，但這時 host B 與 C 所發出的 route discovery packet 就有可能發生碰撞，mobile host D 就收不到從 mobile host B 與 C 送出的 route discovery packets，mobile host A 就找不到正確的路徑將 packet 送到 mobile host D 了，它就必須重新執行廣播 route discovery packet 去尋找新路徑，這就再次浪費網路資源。

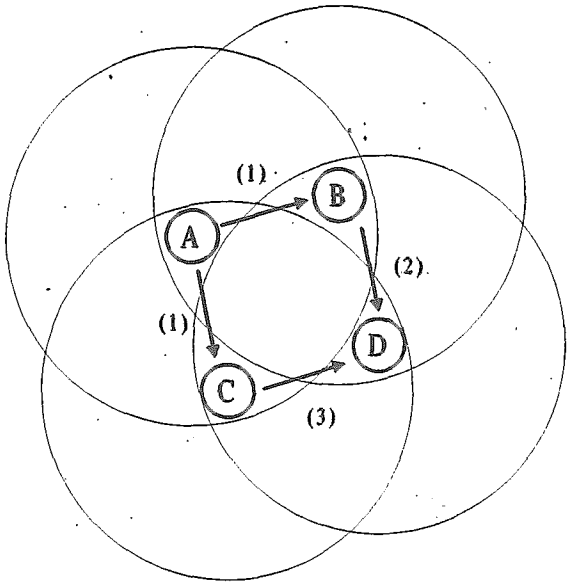


圖 5.2 可能發生 Route Discovery packet 碰撞的環境

5.2.2. 測試方法

在我們實驗的環境中，如圖 5.2，Pentium II 350 notebook 為 mobile host A，Pentium 200 MMX notebook 為 mobile host B，Pentium 233 MMX notebook 為 mobile host C，host D 為 Pentium 200MMX PC。Mobile host A 先廣播一個 route discovery packet，mobile host B 與 C 收到 host A 的 route discovery packet 之後也各自立即將這個 route discovery packet 再廣播出去，然後我們觀察 host D 是否有正確收到從 mobile host B 與 C 的 route discovery packet。

	Packets from B and C	Only packet form C	Only packet from B	No packets received
3000	93	5.7	0.4	0.9
3500	86.5	12.4	0.1	1.0
4000	74.9	0	22.3	2.8
4500	58.6	0.8	40.5	0.9
5000	44.4	50.0	0.3	5.3
5500	47.0	3.0	47.9	2.1
6000	56.1	34.2	6.8	2.9
6500	72.1	25.0	0.7	2.2
7000	83.8	0.1	15.9	0.2

表 5.2 mobile host D 收到的廣播封包統計數字

最初我們發現 host D 總是先收到 host C 的廣播封包接著才收到 host B 的廣播，我們認為是 mobile host C 的配備較高級，所以其處理與傳送封包的速度較快。所以我們在 mobile host C 的程式中要廣播封包之前加入一個空的 for 迴圈，我們發現迴圈行次數約 5000 次就會使得 host D 收到不同順序的廣播或收不到正確的廣播（即碰撞發生），這時可將 mobile host B 與 C 看成兩部性能差不多的機器。當我們將迴圈次數提高或降低，碰撞的次數就減少了。表 5.2 是 mobile host C 在不同的 for 迴圈延遲廣播的情況下，mobile host A 執行 100 次廣播，我們將 mobile host D 所收到從 mobile host B 與 C 的廣播封包做一個統計，圖中的數字是最後的平均結果，mobile host D 只收到從 mobile host C 或 D 送出的廣播封包，甚至是完全沒收到，就表示有碰撞情況發生。我們分別採用了 Lucent WaveLan PCMCIA (相容於 IEEE 802.11 協定) 與 Lucent WaveLan IEEE PCMCIA (使用 IEEE 802.11 協定) 無線網路卡來測試，發現其結果都差不多。

5.2.3. 討論

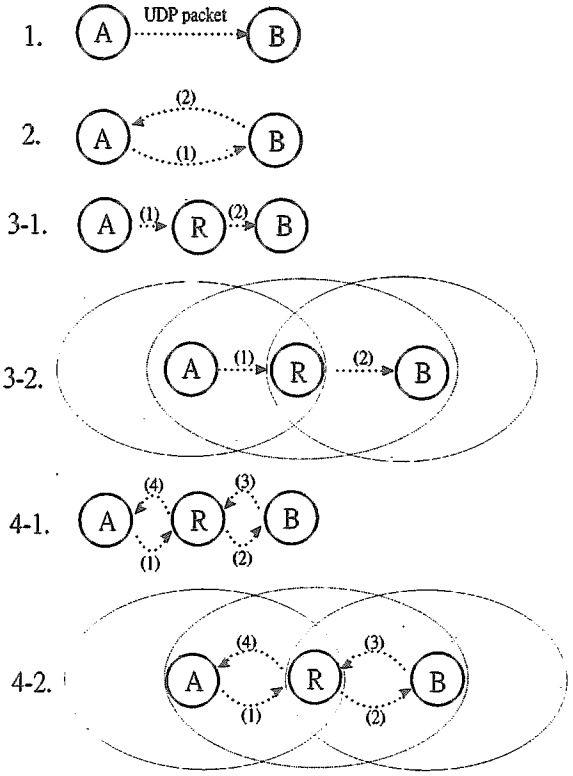
對於 route discovery packet 的碰撞問題，過去已經有一些解決方案被提出。若在一個同質的行動式 ad ho c網路環境中，每一個 mobile host 都是同一型機器而且執行相同的作業系統，所執行程式也差不多，採取[2]已提出防止 route discovery packet 碰撞的方法是相當可行的。但是在一個異質的行動式 ad ho c網路環境中，即網路成員使用了不同的機器與作業系統，就會使得情況變得很複雜，若我們採用[2]的方法，mobile host C 延遲 5 單位時間，mobile host B 延遲 3 單位時間，碰撞情況還是可能發生，而且網路成員愈多，碰撞機率就愈高。這次我們並沒有在設計的繞徑協定中處理 route discovery packet 碰撞的問題，將來我們將針對此一問題進行研究。

5.3. 傳輸速率與傳輸品質的測量

接下來我們量測在行動式 ad hoc 網路環境中的網路傳輸速率，並討論其與繞徑協定之間的關係，以得知傳輸路徑長短與傳輸速率的關係。我們設計了幾種環境如圖 5.3，在狀況 1、2、3-1 與 4-1 的環境中，每一個 mobile host 的通訊範圍均可包含其它的 mobile host，我們測試時使用 UDP 協定連續傳送 10,000 筆資料長度為 1000bytes 的資料，我們忽略 RTS/CTS frame、MAC 層中的 ACK frame 與碰撞後重新傳送的情形，只計算在接收端收到的 UDP 封包數目，粗略估計資料傳輸率，得出的數據如表 5.3。

在圖 5.5 的狀況 1 中 mobile host B 所估算的資料傳輸率約在 1.725Mb/sec，若加上 MAC 層的其它的 frame，傳輸率也逼近 2Mb/sec 了。狀況 2 中 mobile host A 與 B 各自算出資料傳輸率約比單向傳輸時的一半還少一些。狀況 3-1 中，三個 mobile host 可兩兩互相通訊，3-2 中的 mobile host A 與 B 必須透過其它 host 代為轉送封包，這兩種狀況測出的結果差不多，沒有封包遺失的情形。狀況 4-1.中，三個 mobile host 可兩兩互相通訊，分別會因碰撞而短收兩千多筆封包，總傳輸時間也拉長，總傳輸率只有 0.69Mb/sec，4-2.的情況更不理想，這也可能是 frame 因網路卡來不及處理而被放棄。所以在 wireless ad hoc 網路環境中，要求資料的可靠度是最重要的，由於受限於 mobile ad h oc環境的不可靠，所以效能反而不是最重要的考量。

接下來討論我資料傳輸與繞徑協定之間的關係，case 3-1 中 mobile host A 與 B 是可以直接通訊，但是他們卻沒有在繞徑表中將對方設定成鄰居，所以它們傳送的資料得靠其它 mobile host 代為轉送，所以繞徑協定能警覺鄰居的存在而將 case 3-1 轉變成 case 1，情況將大為好轉。Routing protocol 能將 case 4-1 轉成 case 2 的情況也是一樣的理由。所以繞徑協定能夠立即反應網路狀況而找出較短的路徑，這將會大幅改善 ad hoc網路的效能。



5.3.測試傳輸速率的環境

	(1)	(2)	(3-1)	(3-2)	(4-1)	(4-2)
Packets received by A		10000			7932	3672
Packets received by B	10000	10000	10000	10000	7601	5363
Mb/sec evaluated by A		0.793			0.341	0.144
Mb/sec evaluated by B	1.725	0.796	0.879	0.877	0.349	0.159
Total Mb/sec	1.725	1.589	0.879	0.877	0.690	0.303

表 5.3.傳輸速率之數據

6. 結論與未來研究方向

本篇論文的重點，在於實作一個行動式 ad hoc網路環境中的動態繞徑協定，並對此協定進行各種測試。在實作中，我們設計了幾組系統呼叫；利用這些系統呼叫，我們可以很快的實作出 next-hop 型的動態繞徑協定。經過一連串的測試與評估後，未來我們也將考慮把此類之繞徑協定完全實作在作業系統的核心中。

相較於有線網路與有基地台的無線網路，行動式 ad hoc 網路的操作環境是比較嚴苛且更具挑戰性的，我們將設計其他的實驗以挖掘新的問題，進一步提出改進效能的演算法，並且將其實作於我們的繞徑協定中。

7. 誌謝

本研究由以下國科會研究計畫補助：

NSC 88-2213-E-008-013

NSC 88-2213-E-008-014

參考文獻

[1] National Science Foundation. "Research properties in wireless and mobile communications and networking," Report of a workshop held March 24-26, 1997, Airlie, House, Virginia. Available at http://www.csie.nsf.gov/anir/ww.html.

[2] Jose Broch, David B. Johnson, and David A. Maltz, "The Dynamic Source Routing for Mobile Ad Hoc Networks," Internet-Draft, draft-ietf-manet-dsr-01.txt, December 1998, Work in progress.

[3] Zygmunt J. Hass and Marc R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet-Draft, draft-ietf-manet-zrp-00.txt, November 1997, Work in progress.

[4] David B. Johnson, "Routing in ad hoc networks of mobile hosts," In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 158-163, December 1994.

[5] David B. Johnson and David A. Maltz, "Dynamic Source routing in ad hoc wireless networks," In mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Academic Publishers, 1996.

[6] David A. Maltz, Jose Broch, and David B. Johnson, "Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed," Draft. 1999, Work in progress.

[7] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," MobiCom'99, Aug. 1999.

[8] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequence distance vector routing for mobile computers," Computer Communications Review, 24(4):234-244, Oct. 1994.

[9] Charles Perkins, "Ad Hoc On Daemon Distance Vector (AODV) Routing," Internet-Draft, draft-ietf-manet-aodv-00.txt, November 1997. Working in progress.