東海大學資訊管理研究所

碩士學位論文

貪婪式空隙迴避演算法於無線感測網路之路由應用

Greedy Geographical Void Routing for Wireless Sensor Networks

指導教授：姜自強 博士

研 究 生：李莎百 撰

中華民國 101 年 6 月 26 日

東海大學資訊管理學系碩士學位

考試委員審定書

資訊管理學系研究所 李莎百 君所提之論文

貪婪式空隙迴避演算法於無線感測網路之路由應用

Greedy Geographical Void Routing for Wireless Sensor Networks

經本考試委員會審查，符合碩士資格標準。

學位考試委員會 召集人：侯廷偉 (簽章)

委 員：

中 華 民 國 101 年 06 月 26 日

誌謝

首先感謝家人的支持，父母從小陪伴著我們學習與成長，並讓我們無後顧之憂，不用煩惱學費的問題，可以全心全意的專注於研究上。在碩二上學期時更支持我到國外拓展視野，在義大利實習半年，闖出自己的一片天空。而每當遇到挫折時，家總是我最好的避風港與心靈的歸宿，並在重新整理過後，再出發。感謝好友玉敏與昀臻，在論文衝刺階段的照顧與陪伴。感謝大學同學翊婷、庭安與若俞，友情的支持與鼓勵，讓我有力量能持續努力。

在研究所論文寫作期間，承蒙恩師姜自強教授對學生的栽培與指導。在姜老師不厭其煩的耐心指導下，能在一上時順利完成論文研討會的發表，培養獨立完成研究與寫作的能力。並在一下時著手研討會文章的翻譯，培養我們論文英文寫作的能力。除了學術理論之外，老師讓我們從產學合作中，能在求學期間提早接觸業界進行實作，並將所學整合與應用開發。感謝老師允許並鼓勵我於二上時，暫時放下研究到國外去闖蕩實習半年，完成我的夢想。回國收心後，二下論文最後衝刺階段，在跟老師不斷的反覆討論與校稿與循循善誘下，如期的順利完成碩士論文的寫作。此外老師圓融的待人處事方法與陽光開朗的活力也是我們學習的榜樣，能夠有正向的態度去面對困難與挑戰。

系上碩班同學間感情融洽，感謝班代祐平，為了班上事務勞心勞力的付出；感謝凱琳，在課程分組合作上的好拍檔；感謝淑齡，總是有著很開朗的笑容，與熱心的付出；感謝韋辰，在論文的寫作上互相討論與學習；感謝姿菁，常分享很多資源的訊息給大家；感謝裕欽，收集紀錄了班上點點滴低的回憶；感謝秉昕，帶給大家很多的歡笑；感謝華煌，常不辭辛勞載大夥上山下海與熱心的幫助；感謝柏彥，課外休閒活動的好夥伴；感謝筱婷，在美食與娛樂上的分享。感謝學弟凱毅與翰霖，一起合作老師的研究與計畫。

論文名稱：貪婪式空隙迴避演算法於無線感測網路之路由應用
校所名稱：東海大學資訊管理學系研究所
畢業時間：2012年06月
研究生：　李莎百　　指導教授：姜自強
論文摘要：

隨著無線網路科技的進步與各種不同的應用於現實生活中，其中無線感測網路(Wireless Sensor Network ,WSN)便是近年來熱門的研究領域。由於無線感測網路節點隨時移動與拓樸變化快速的特性，節點間常有收訊不到的空隙問題，節點在傳遞封包時我們無法預測空隙問題會在何時與何地發生,這將會造成封包遺失、重送、重繞...等額外的傳送成本與電力輸出，所以改進無線感測網路的地理路由方法上更顯重要。本研究以發現空隙法為基礎找出空隙後，提出貪婪式空隙迴避法來解決無線感測網路的空隙問題，我們以來源端節點對空隙做兩切線形成通知空隙存在的扇形範圍，再以來源端節點與目的地節點形成的直線與扇形範圍間所產生的夾角來選擇下個相鄰節點進行路由。在瞬息萬變的動態無線感測網路環境中，本研究所提出的迴避空隙方法能更省時且有效率的轉送封包，解決目前無線感測網路的地理空隙問題。

關鍵詞：無線感測網路、網路路由、無線網路、貪婪空隙迴避法、繞過空隙

Title of Thesis：Greedy Geographical Void Routing for Wireless Sensor Networks
Name of Institute: Tunghai University, Institute of Information Management
Graduation Time：（06／2012）
Student Name：Sha-Pai Li Advisor Name：Tzu-Chiang Chiang
Abstract：

With wireless network technology's advancement, there are a variety of applications in our life. Among that wireless sensor network is the popular research area in recent years. As the wireless sensor network node move at any time and the topology fast change feature, nodes often have void problem. When node transports packet, we can't predict void problem will occur in when and where. This will produce packet losing, resending, rerouting, additional transmission cost and electric power output. Thus, how to improve wireless network technology's geographic routing method is important. In this research, we use the find void algorithm to find void. Then we propose a greedy void avoiding algorithm to solve wireless sensor network's void problem. We use source node and void to draw two tangents form the announce void existence's fan range, then use source node and target node to draw a line with the fan range's angle to select next neighbor node for routing. In the rapidly changing dynamic wireless sensor network environment, we expect this research's greedy void avoiding algorithm that we proposed can be more time-saving and more efficient to forward packet. And improve current wireless sensor network's geographical void problem.

Keywords：wireless sensor network, internet routing, wireless network, greedy void avoiding algorithm, bypassing void

目錄

表目錄

圖目錄

第一章 前言

隨著無線網路科技的進步，各種不同的應用於現實生活中，其中無線感測網路(Wireless Sensor Network ,WSN)便是近年來熱門的研究領域。由於無線感測網路節點隨時移動與拓樸變化快速的特性，節點之間常有收訊不到的空隙問題，若無法將需要傳遞的封包送達到目的節點時，將產生額外的傳送成本與電力輸出。節點在傳遞封包時我們無法預測空隙問題會在何時與何地發生，這將會造成封包遺失、重送、重繞...等額外的傳送成本與電力輸出，所以如何改進無線感測網路的地理路由方法更顯重要。

貪婪演算法是一個被廣泛應用的方法，因其具有低複雜度和良好的可延展性。由於貪婪演算法的高效性以及其所求得的答案比較接近最佳結果，貪婪演算法也可以用作輔助演算法。然而，貪婪演算法並不是總是可以解決問題的。當網路中存在空隙時，使用貪婪演算法可能導致空隙周邊產生阻塞區域，並降低網路生命週期和有效流量。

通訊空隙形成的原因為，當發送節點未能找到下一個在其附近有效節點到達地理目標節點的現象，稱為通訊空隙。由於在動態無線網路環境中不可預測的節點部署模式中，無法預測何時何地會出現一個空隙區域。如果沒有適當的空隙處理技術到位，某些封包可能會在網路中遺失，浪費寶貴的網路資源以及阻礙無線網路節點對節點之間的通信。

通訊空隙是一個具有挑戰性的地理路由問題，為了能有效的利用地理路由在下一代的無線網路中，此問題必須解決。雖然某些密集部署的無線節點可以減少在通信空隙網路中發生的可能性，但一些數據封包仍然可能遇到障礙空隙的誘導與無線網路不可靠的邊界節點等等。即使有一條貪婪式轉發的有效路徑可到達目標節點，這些數據封包仍然可能被丟棄。因此，必須設計一個有效和高效方式的地理路由空隙處理技術。

地理轉發工作有兩種模式：地理貪婪轉發模式和空隙處理模式。在貪婪轉發模式，選擇下一節點的封包轉發是根據位置的當前節點、對周邊節點和目標節點。如果發送

者需要傳送封包時，而封包的目標節點無法找到有效的下一節點，它將切換到空隙處理模式。在這種模式下，節點試圖由空隙區域的周圍選擇路徑，因為它很可能是一個從來源端節點到目的地節點間有效的路徑。

本研究以發現空隙法為基礎找出空隙後，提出貪婪式空隙迴避法來解決無線感測網路的空隙問題，我們以來源端節點對空隙做兩切線形成通知空隙存在的扇形範圍，再以來源端節點與目的地節點形成的直線與扇形範圍間所產生的夾角來選擇下個相鄰節點進行路由。在瞬息萬變的動態無線感測網路環境中，本研究所提出的迴避空隙方法能更省時且有效率的轉送封包，解決目前無線感測網路的地理空隙問題。

第二章 文獻探討

地理路由最初是以封包無線網路在 20 世紀的 90 年代被提出。是以移動設備或裝置的位置作為基礎的路徑選擇演算法，基於所在地為基礎或基於方向為基礎的路由演算法。近年來，隨著全球定位系統的應用和自身定位配置機制的進展下，又重新成為熱門的研究議題。因為它提供了下一代無線網路更具吸引力的訊息傳遞解決方案，例如移動隨意無線網路(MANETs)、車輛移動隨意無線網路(VANETs)、無線感測器網路(WSNs)和無線網狀網路(WMNs)。

第一節 無線感測網路

無線感測網路(Wireless Sensor Networks)是由一到數個無線資料收集器(Wireless Data Collector)以及為數眾多的感測器(Sensor)所構成的網路系統，採用無線通訊(Wireless Communicate)的方式在元件之間進行通訊。因此，我們可以任意擺放感測器或是無線資料收集器，不但可以省下可觀的佈線費用，且帶來極大的便利。

在無線感測網路的架構下，感測器的設計是以省電、體積小、價格低廉且具有感應環境裝置為目標，感測器本身就像是一台小型電腦，並配備了簡單的感測

裝置、運算裝置和無線傳輸裝置。感測裝置可以針對環境中我們所感興趣的事物(如溫度、光源等)做偵測行為，並將所收集的資料先做簡單運算處理後，再透過無線傳輸裝置，將資料回傳給資料收集器。最後，我們就可以根據資料收集器所收集的資料，了解環境的狀態，進行開發與應用[7,8,11,13,14,22,26,32,33]。

第二節 無線感測網路的網路拓樸

一般感測網路的建立與維護可細分為下面三個階段：

壹、 事先規劃與實際部署階段 (Planning and Deployment Phase)

感測器的部署可以事先精心規劃，再依據所規劃的藍圖逐一擺放感測器；然而在某些特殊環境下，例如：海洋、戰場，或是森林等，感測器的佈置是無法預測的，因此在這種情況下，感測器往往以非常大量的方式，透過船隻、飛機，或是其他機械等，隨意散佈在感測環境中。

貳、 後部署階段(Post-deployment Phase)

理論上，在感測器部署完成後，就已經形成一個完整的無線網路架構。但是，感測器的位置容易受到環境因素，例如水力、風力或是人為移動等因素而改變，或是當感測器電力減弱或故障損壞時，可能會形成空隙，進而造成網路拓樸(Network Topology)的改變；此時系統必須對拓樸的改變做出即時的反應，並迅速對不完整的網路拓樸做出適當的修正。

參、 新增感測器階段(Redeployment of Additional Phase)

在經過一段時間後，許多感測器可能由於電力嚴重不足或是損壞的緣故，造成整個感測網路中，可以偵測環境以及傳輸資料的節點個數不足。此時，系統已經無法自動對整個網路架構做出修補的動作，所以，這時候必須依靠外力(人員、機械等)，在這個網路中新增一些額外的感測器以彌補感測器數量不足的問題。有了這些新增的感測器之後，原先不完整的網路彷彿注入了新生命，可以再自動重

新組態(Auto-configuration)網路架構，以繼續執行任務。

第三節 貪婪路由演算法

地理路由通常以所謂移動設備或裝置的位置為基礎的路徑選擇演算法，基於所在地為基礎或基於方向為基礎的路由演算法，最初是以封包無線網路在 20 世紀的 90 年代被提出。不同於拓樸路由，地理路由是利用地理位置資訊，而不是節點間拓樸連接資訊來傳遞數據封包，並逐步接近後，最終到達目的地節點。在大多數的地理路由演算法中,只有 one-hop 相鄰的鄰居節點地理資訊會被使用到。因此，地理路由演算法中不需要建立或維持從來源節點到目的節點的完整路線。中間節點無需儲存路由表，所以沒有必要傳輸路由資訊來更新路由。運用節點區域的運作的功能使得地理路由變得簡單，且具有可擴展性。地理路由還使用地理建設服務，支援封包傳遞到所有指定的地理區域之節點。

貪婪演算法是一種在每一步選擇中，採取當前狀態下最好的選擇，進而希望導致結果是最好的演算法。此演算法在有最佳子結構的問題中最為有效，最佳子結構的意思是局部最佳解能決定全局最佳解，如圖 2-1。簡單地說，問題能夠分解成子問題來解決，子問題的最佳解能遞推到最終問題的最佳解。地理路由主要依賴於一個非常簡單的地理貪婪轉發機制，每當封包轉發到離目的地最近的相鄰節點，也就是本地的最佳下一節點時，積極的尋找最合適的傳遞路徑到達目標節點，並且避免產生一個循環路徑。在選擇轉發封包的下一節點是根據下一節點與當前節點位置、下一節點與周邊節點和下一節點與目標節點。一個節點可以決定自己的位置，無論是預先配置或該節點是固定的，也可通過全球定位系統接收器或通過定位算法。

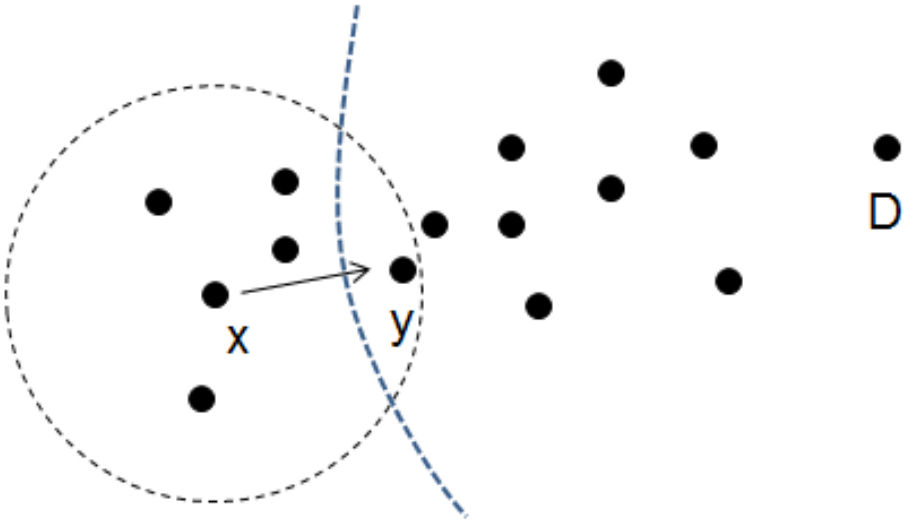


圖 2-1 貪婪演算法範例

貪婪演算法是一個被廣泛應用的方法,因其具有低複雜度和良好的可延展性。由於貪婪演算法的高效性以及其所求得的答案比較接近最佳結果，貪婪演算法也可以用作輔助演算法。然而，貪婪演算法並不是總是可以解決問題的。當網路中存在空隙時，使用貪婪演算法可能導致空隙周邊產生阻塞區域，並降低網路生命週期和有效流量。例如，如果所有的相鄰節點相對發送節點都是遠離目標節點，發送節點無法找到下一個在其附近的有效節點到達地理目標節點。這也是本研究可以進行研究，並予以改善的地方[14,15,16,17,20,27,28,29,30,31]。

第四節 通訊空隙

在近年文獻的調查中顯示，這些通訊空隙分類處理技術分為六個類別，分別設計了不同的方法，其分類為 1.基於平面與圖形路由 2.幾何路由 3.氾濫式路由 4.基於成本考量路由 5.啟發式路由和 6.混合路由等等。

無線網路中通訊空隙對於在地理式貪婪轉發路由協定下，若無法將需要傳遞的封包送達到目的節點時，將是一個重要的議題。通訊空隙形成的原因為，當發送節點未能找到下一個在其附近有效節點到達地理目標節點的現象，稱為通訊空

隙，如圖 2-2。通常也被歸納為區域最大化現象或區域最小化現象。其中空隙的部份也區分為二：開放式空隙與封閉式空隙。

通訊空隙是一個具有挑戰性的地理路由問題，為了能有效的利用地理路由在下一代的無線網路中，此問題必須解決。雖然某些密集部署的無線節點可以減少在通信空隙網路中發生的可能性，但一些數據封包仍然可能遇到障礙空隙的誘導與無線網路不可靠的邊界節點等等。即使有一條貪婪式轉發的有效路徑可到達目標節點，這些數據封包仍然可能被丟棄。因此，必須設計一個有效和高效方式的地理路由空隙處理技術。

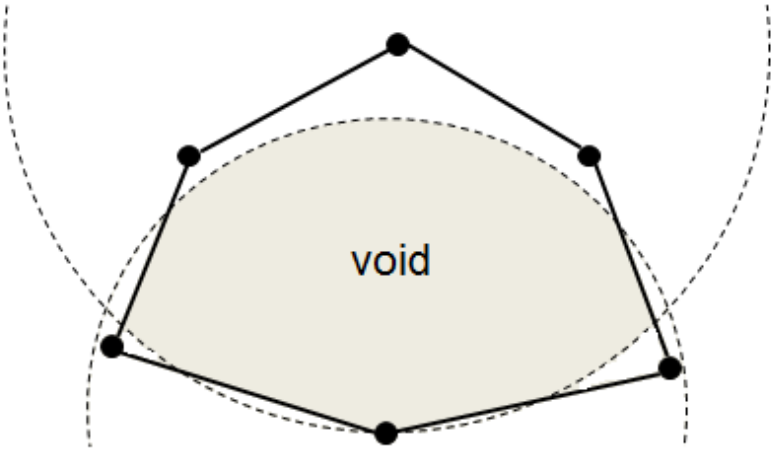


圖 2-2 空隙問題

地理路由通常有兩個主要因素：位置服務和地理轉發策略。在來源節點發送數據封包之前，位置服務是負責確定其數據封包目的地節點的位置。封包的位置附帶在封包標頭中，使中間節點可以學習該數據封包轉發往何處。地理轉發工作有兩種模式：地理貪婪轉發模式和空隙處理模式。在貪婪轉發模式，選擇下一節點的封包轉發是根據位置的當前節點、對周邊節點和目標節點。一個節點可以決定自己的位置，無論是預先配置的固定節點、通過全球定位系統接收器，或通過定位算法。在轉發過程中找到一個更好的選擇節點時，可以選擇更新定位再轉發封包標頭。

如果發送者需要傳送封包時，而封包的目標節點無法找到有效的下一節點，它將切換到空隙處理模式。在這種模式下，節點試圖由空隙區域的周圍選擇路徑，因為它很可能是一個從來源端節點到目的地節點間有效的路徑拓樸。

由於在動態無線網路環境中不可預測的節點部署模式中，無法預測何時何地會出現一個空隙區域。如果沒有適當的空隙處理技術到位，某些封包可能會在網路中遺失，浪費寶貴的網路資源以及阻礙無線網路節點對節點之間的通信。最簡單的空隙處理技術為氾濫式廣播，當傳送封包遇到一個空隙節點時，所有節點首次皆收到滯留封包，並執行轉播。如果至少存在一個路徑時，氾濫式廣播技術的確可以將該封包傳遞到達目的地節點。但這種方法在資源利用的效率方面，效果非常差。因為網路所有其他的節點將封包轉發一次，目的地節點可能會收到太多從不同路徑，且不必要的相同封包。如果沒有下一個有利路徑進展的節點時，應該將數據封包轉發到發生在空隙節點或末路節點上的節點，但它可能導致循環路徑的問題[2,5,6,9,12,18,19,23,24,25]。

第五節 避免空隙路由

避免空隙路由的問題，至今已有許多不同角度的觀點。Karp 學者提出了 Greedy Perimeter Stateless Routing(GPSR)，是一結合貪婪轉發機制和右手周長定則的路由方法，如圖 2-3。GPSR 是一個可擴展且完全分佈的協定，並保證封包的有效傳輸。然而，在右手周長定則路由中，封包沿著空隙的表面路由，因此於節點上產生嚴重的負載[1]。Yu 學者藉由構建虛擬範圍圍繞網路空隙解決空隙定位的問題。一個於虛擬範圍內適當的中心節點被選擇為對每個封包傳輸進行重繞。他們提出的另一種方法，是建構一個橢圓形來圍繞空隙。當封包傳輸到橢圓範圍時，在繼續貪婪轉發之前，先改變方向並沿著切線方向前進[30]。

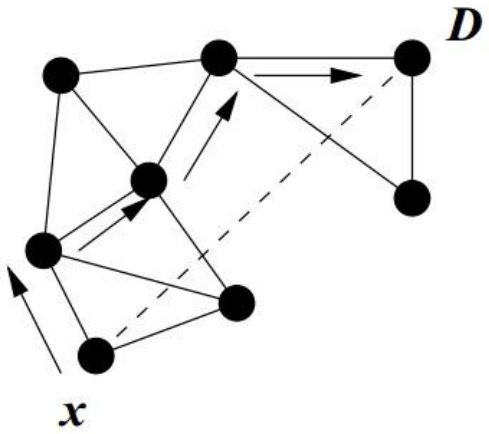


圖 2- 3 Karp 學者所提出 GPSR 周長轉發範例[1]

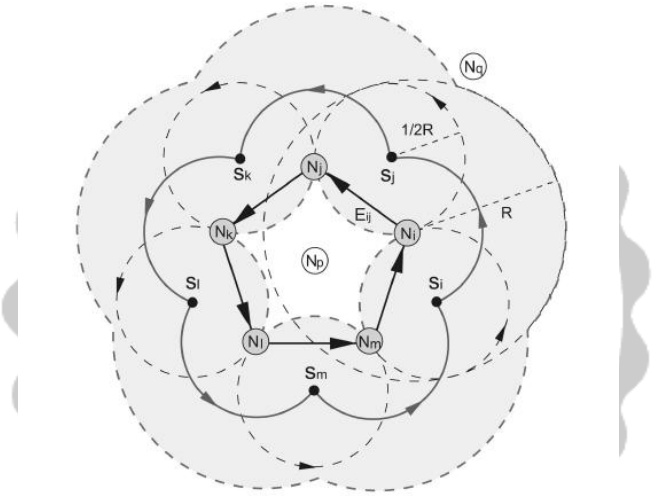


圖 2- 4 Liu 學者所提出 GAR RUT 方法[4]

Ye 學者進一步藉由改善建置動態重繞路由協定，為了分散流量和延長網路生命週期。Aissani 學者發展出一個目的在於減少點對點的傳輸延遲方法。空隙的中心是被計算且宣告給在空隙區域範圍內的所有節點。之後節點利用這些資訊來選擇轉發範圍和事先避免空隙[3]。假設基於 UDG 的基礎上，Liu 學者發展 greedy anti-void routing (GAR) 演算法，目的在於提高路由效率，並保證封包的傳輸。使用以節點傳輸範圍半徑的虛擬滾動球，如圖 2-4，GAR 是藉由減少所需節點轉發次數來繞過空隙[4,10,21]。

第三章 貪婪式空隙迴避演算法

本研究計畫目的旨在解決問題

1、 有效迴避空隙問題

發現並定義空隙後，有效避免空隙，解決目前無線感測網路的地理空隙問題。

2、 減少資料傳遞成本

減少轉發次數，縮短路由距離，能更省時且有效率的轉送封包。

3、 降低能源的輸出

減少傳遞重複的資料，降低網路流量以達到節省能源的目的。

並提出對應的解決方法步驟

1、 發現空隙

2、 通知空隙

3、 選擇相鄰節點

無線感測網路環境假設

1、 模擬環境為靜態路由

2、 目的地節點為匯集點

3、 尋找空隙的額外成本

在無線感測網路環境中，通訊空隙的問題一直是個不確定因素，隨時可能在資料傳遞上產生問題。當無線感測網路環境中有障礙物例如湖泊與建築物的阻礙，便會有通訊空隙的問題產生。因此，如何能有效的解決節點間常有收訊不到的空隙問題，有效提昇傳輸的品質就顯得相當的重要。雖然在發現空隙與繞過空隙的過程中，會產生比一般路由額外的尋找空隙成本，但能有效的避免空隙問題，提昇在有通訊空隙下網路傳輸的效能。本研究所提出的貪婪式空隙迴避法包含三個步驟：發現空隙、通知空隙和選擇相鄰節點。

第一節 發現空隙

當在通訊範圍內沒有足夠的感測器時，便會產生通訊空隙，而封包的傳輸將會有不可預測的情況產生。而什麼是空隙、如何定義空隙和如何繞過空隙，便是我們所要探討的。首先定義出現存的空隙分佈情形，空隙是由當封包無法繼續往目的地傳遞時所產生卡住的節點與空隙邊緣節點所形成。有了這些卡住節點的資料後，將節點進行串連得到空隙的邊緣，進而得到空隙範圍。

傳輸節點在傳輸時，因空隙問題而無法繼續前進時，該節點則視為卡住節點，我們能藉由演算法找出包含卡住節點及其他也是在空隙邊緣上的所有因空隙而無法繼續傳輸前進的卡住節點，並將之連結形成該範圍內的封閉空隙範圍。

$$y = m \cdot x + c \quad (1)$$

$$y = n \cdot x + c \quad (2)$$

$$m \cdot n = -1 \quad (3)$$

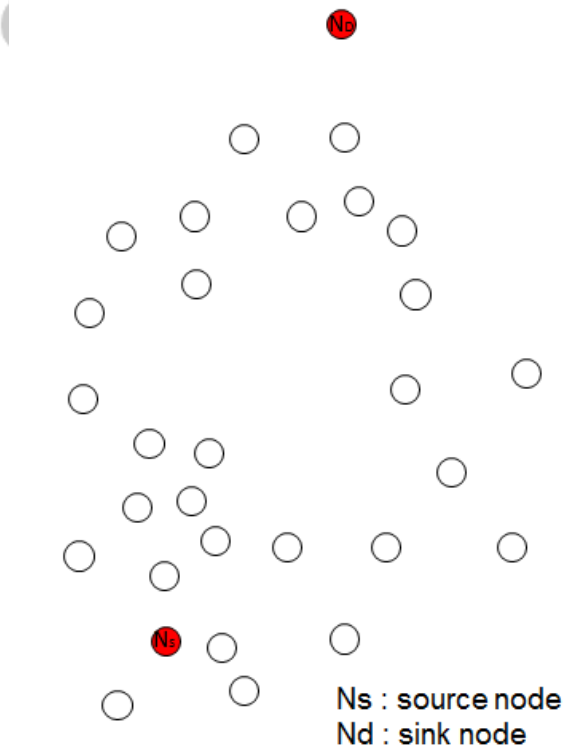


圖 3- 1 定義初始節點與目的地

首先定義初始節點與目的地節點，如圖 3-1。以傳輸節點和目的地節點的直線，畫出水平垂直線(1) (2)，垂直線斜率互相乘積為-1 (3)，如圖 3-2。以水平線為基準，尋找下一傳輸範圍內空隙邊緣節點。而傳輸範圍內節點可分為兩種，當節點位於傳輸範圍內，向後尋找的節點，則為卡住節點。而當節點位於傳輸範圍內，向前尋找節點，則為空隙邊緣節點。

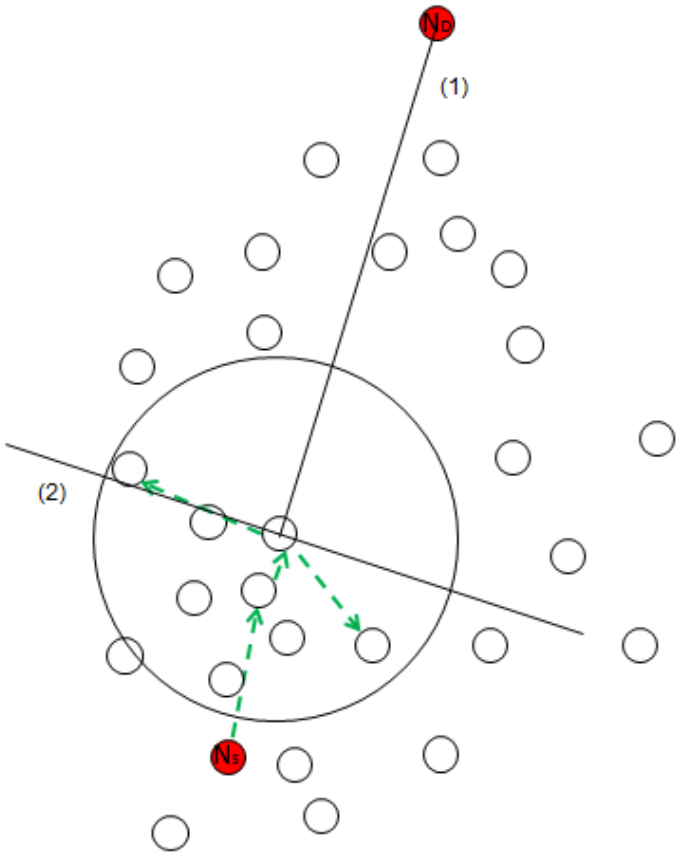


圖 3- 2 選擇空隙邊緣節點

在選擇下一節點時我們會選擇在傳輸範圍內，如果是從左路尋找時，依水平線左邊開始以左手定則尋找下一空隙邊緣節點。反之，從右路尋找時，依水平線右邊開始以右手定則尋找下一空隙邊緣節點。最終回到初始卡住節點，形成循環路徑後變停止，產生一組空隙邊緣節點。

當傳輸範圍內有兩個節點正好角度相同時，由一開始卡住的節點向左右兩路出發，分別依左右手定則找出空隙邊緣節點，最後兩路進行交會後，完成連結後形成空隙範圍。

封包沿著邊緣轉發，追加新的空隙邊緣節點。當傳遞結束又返回到初始卡住節點後,每個邊緣節點的位置會被編碼於發現空隙封包,建立一組空隙邊緣節點。再從空隙邊緣節點中選擇兩節點，而兩點之間的距離是空隙邊緣節點中任意兩點間距離最長的，並計算線段的中點 v，如圖 3-3。

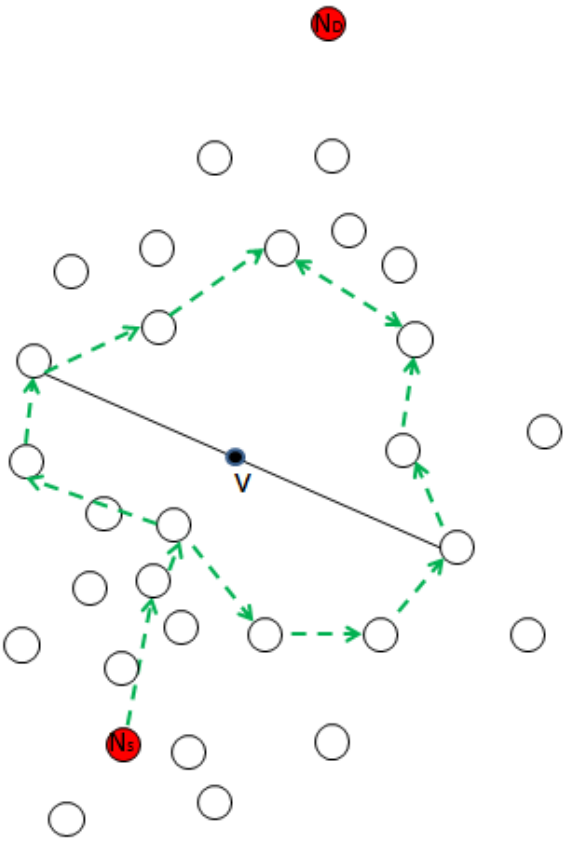


圖 3- 3 定義空隙範圍

set node N_i (N_i , (x_i,y_i),s ←1=stuck node,0=not stuck node) //節點資訊

set R to the value of node transmission radius //設定傳輸半徑

set source node N_s(x_s,y_s) and destination node N_d(x_d,y_d) //設定來源節點與目的地節點座標

set n=0 //設 n 為一常數

set SBN to the set of void boundary node ←SBN={N_1,N_2,…….N_{17},N_{18}} //設定一組空隙邊緣節點

if (node N_p is stuck node) //完成一組空隙邊緣節點

then

start to find next void boundary node N_b;

for(n=0;n++; next void boundary node return to the stuck node N_p){

draw the horizontal and vertical lines ←$y = m \cdot x+c$ with $m = (y_d - y_{Ni})/(x_d - y_{Ni})$;

case1:stuck node

backward to find next void boundary node(the third quadrant and the forth quadrant);

case2:not stuck node

forward to find next void boundary node(the first quadrant and the second quadrant);

cross(Point& o, Point& a, Point& b) { //向量 oa 與向量 ob 進行外積，判斷 oa 到 ob 旋轉的方向。

return (a.x - o.x) * (b.y - o.y) - (a.y - o.y) * (b.x - o.x);

}

else: the angle is the same

intersect(Segment& s, Point& p) { //外積判斷點與線段是否共線，內積判斷點位於線段之間。

return cross(s.p1, s.p2, p) == 0 && dot(p, s.p1, s.p2) <= 0;

}

select which node's slope is max ← $m_{max} = (y_{Ni} - y_{Np})/(x_{Ni} - y_{Np})$;

add node N_b to the set of SBN;

if (next == start) break; //回到初始節點，完成空隙

}

第二節 通知空隙

在找到空隙之後，定位了空隙中心點 v 的座標為 x_v 和 y_v，並透過 sink 目的地節點負責讓所有在通知範圍內的節點知道空隙的範圍。有了空隙範圍的位址，再通知範圍內的節點，即可在遇到空隙前能事先採取行動來迴避空隙，如圖 3-4。

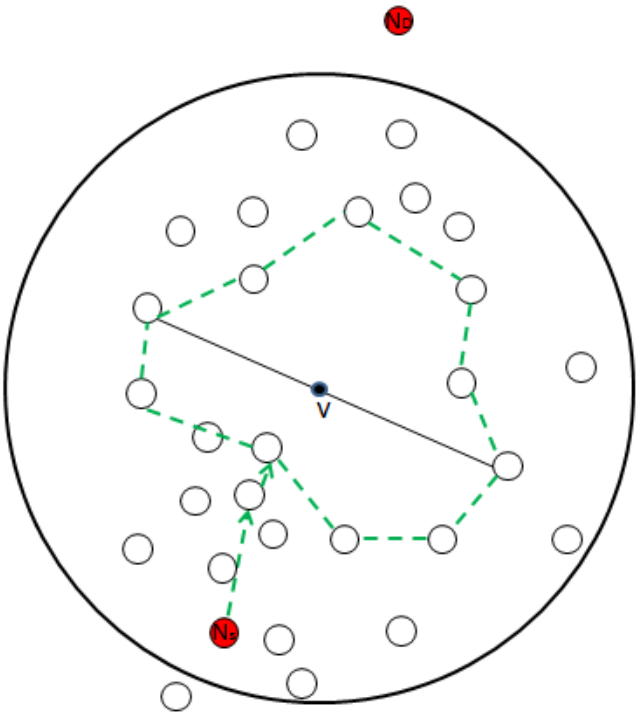


圖 3- 4 空隙通告範圍

當節點座標與空隙中心距離小於通知範圍半徑時，該節點位於空隙範圍內(4)。

$$(x_v-x_i)^2+(y_v-y_i)^2<R^2 \qquad (4)$$

通知範圍半徑 R 是由無線感測傳輸範圍 SR 和空隙的半徑 r 所產生的公式(5)所估計出。

$$R= r+n/2\times SR, \text{ with } n= 1, 2, 3\ldots \qquad (5)$$

之後再以包覆空隙為前提下，取空隙邊上的兩點分別畫切線形成有角度的空隙迴避範圍。分別延伸兩條切線與原通知範圍的圓周相交。相交後產生 j 和 k 兩點，形成一個新的扇形空隙迴避範圍，如圖 3-5。

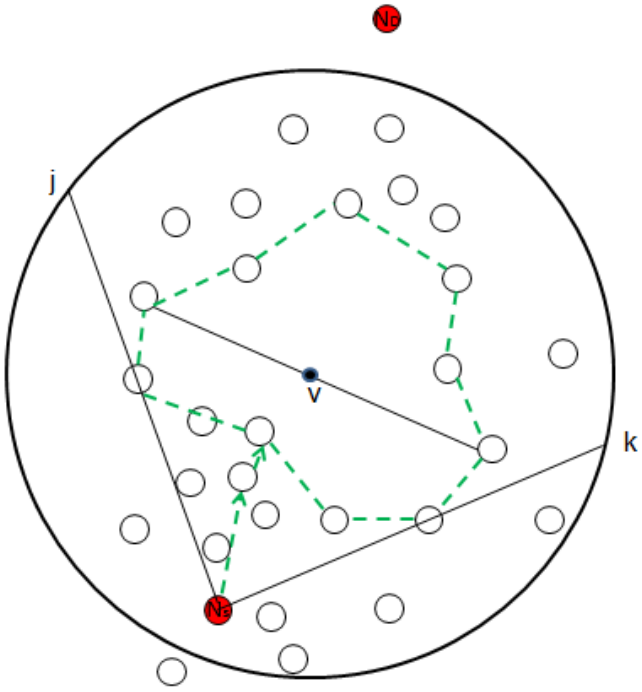


圖 3- 5 新的扇形空隙迴避範圍

由於無線感測網路節點有隨時移動與拓樸變化快速的特性，原有的空隙可能更大或是改變形狀。因此，負責通知空隙範圍的目的地 sink 節點會週期性的重新發送發現空隙封包和重新執行發現空隙步驟來更新發現空隙封包。如果空隙有任何的改變，在通知範圍內的所有節點將在下一次更新時得到通知。

```
if (finish the set of SBN)
while (SBN≠ ∅){
    Farthest Pair ← Rotating Caliper(find 2 nodes which distance is the longest in SBN)
    get void center point(xv, yv) ← ((xp1+xp2)/2, (yp1+yp2) /2);     //找出空隙中心座標
    AN ← { an | an ∈ (xv-xi)2+(yv-yi)2<R2 , R= r+n/2×SR, with n= 1, 2, 3…}    //空隙通知範圍
    Tangent Line ← Sweep Line ( find 2 nodes which can cover void in SBN )    //包覆空隙的切線
}
```

第三節 選擇相鄰節點

在即時的地理路由協定下，節點轉發封包到下一個相鄰節點中，節點選擇區域為轉發範圍。轉發範圍是節點 s 到達目的地節點 d 的轉發相鄰節點範圍。

當節點 s 接收到要轉發給目的地節點 d 的封包時，會先掃描轉發範圍。s 節點會執行以下的規則：

傳輸節點 s 和目的地節點 d 的直線 $\overline{sd}$ 與 $\overline{sj}$ 和 $\overline{sk}$ 兩切線相交產生兩夾角後，選擇夾角角度較小的一邊(6)。如果該夾角是靠傳輸點和目的地節點連線的左邊，則以逆時針搜尋轉發範圍內相鄰節點。如圖 3-6，$\overline{sj}$ 與 $\overline{sd}$ 間之夾角比 $\overline{sd}$ 與 $\overline{sk}$ 間之夾角角度較小。以傳輸節點 s 和目的地節點 d 的直線 $\overline{sd}$ 開始逆時針方向從轉發範圍內選擇轉發相鄰節點，並選擇最靠近切線的節點，作為下一傳輸節點。圖中相鄰節點 m 為以逆時針尋找最靠近切線的節點，故作為下一個轉發的節點。反之，如果是靠傳輸點和目的節點連線的右邊，則以順時針搜尋相鄰節點。

$\angle dsj \quad < \quad \angle dsk$ (choose smaller side)　　(6)

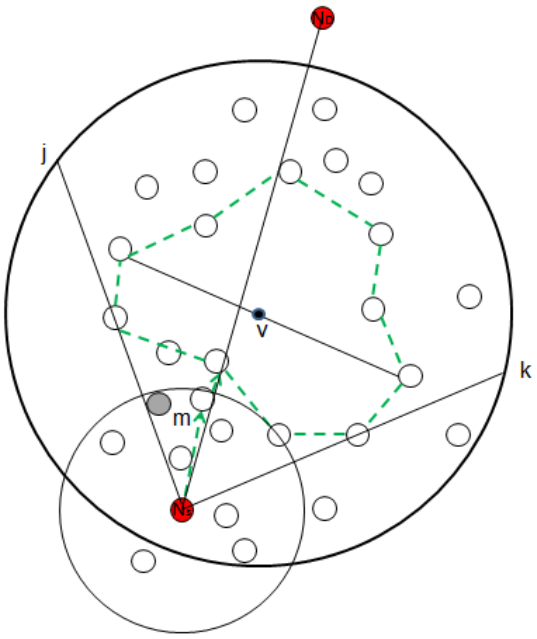


圖 3- 6 相鄰節點的選擇

在貪婪前進演算法中，當傳輸節點想要傳輸封包到目的地節點，他會從傳輸範圍中選擇節點中有最短路徑到目的地節點，且比該節點到目的地節點距離短的節點作為下一節點。

而在轉發的過程中，如傳輸節點 s 和目的地節點 d 的直線$\overline{sd}$已不再包覆在扇形空隙迴避範圍內時，便回歸到使用貪婪式地理路由演算法機制進行傳遞。將封包轉發到離目的地最近的相鄰節點，也就是本地的最佳下一節點時，積極的尋找最合適的傳遞路徑到達目標節點，並且避免產生一個循環路徑。

```
Case 1: void mode
compare_angle ←   find ∠θ1  <  ∠θ2    //選擇夾角角度較小的一邊
cross(Point& o, Point& a, Point& b) {   //向量 oa 與向量 ob 進行外積，可以判斷 oa 到 ob 旋轉的方向。
   return (a.x - o.x) * (b.y - o.y) - (a.y - o.y) * (b.x - o.x);
}
N ← find the node m ∈ N with minimal dist(m, Tangent Line)       //選擇靠近切線的節點
Case 2: Greedy forwarding mode
M is the current node.
D is the destination node.
M ← S
while (M≠D){
     N ← { n | n ∈ M's one hop nodes}
     If (N≠Ø){
          find the node n ∈ N with minimal dist(n,D)   //節點到目的地節點距離短的節點作為下一節點
          M ∈ n
     }
}
```

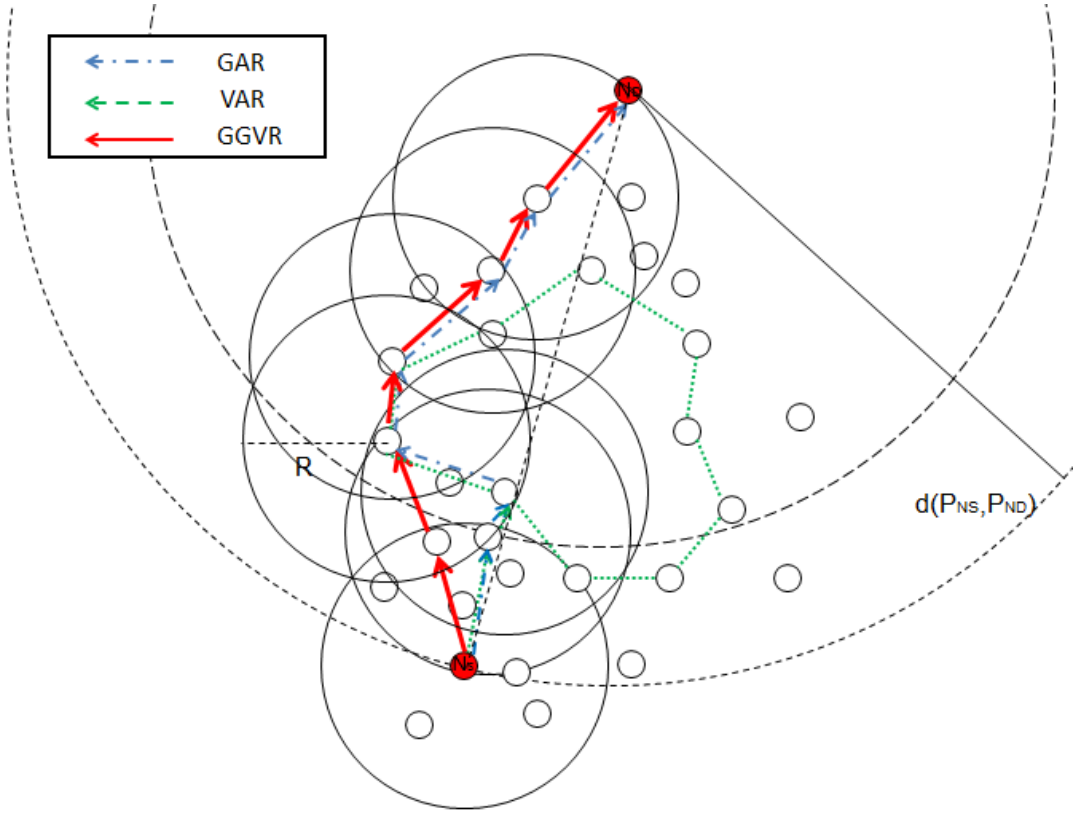


圖 3-7 演算法路由比較

圖 3-7 為本演算法 GGVR 與 GAR 的路由模擬演繹圖，本研究方法預期能在發現並定義空隙後，有效避免空隙，並減少轉發次數，縮短路由距離，能更省時且有效率的轉送封包，改進目前無線感測網路的地理空隙問題，提昇網路路由效能。

第四章 模擬實驗結果

第一節 網路劇本產生器(Network Scenario Generator)

NSG 是一個專為 NS2 所設計的劇本產生器，使用 NSG2 時，主要分成五個模式，這些模式大多都是依照 NS2 中主要的幾種物件來設計。NSG2 中包含以下五種模式：

1. Hand mode
2. Node mode
3. Agent mode
4. Application mode
5. Parameters

壹、 手動模式(Hand mode)

在 Hand mode 模式下的功能最為單純，主要是讓使用者可以調整要觀看的範圍。當建置一個大型的劇本時，可能會包含大量的 Node，畫面可能容納不下所有的物件，這時可利用 Hand mode 來調整要觀看的範圍，使用時只要按住滑鼠的左鍵，然後拖曳即可。

貳、 節點模式(Node mode)

在 Node mode 模式下，使用者可配置 Node 的位置，在設計無線網路劇本時，Node 的座標就變的非常重要，因為每一個 Node 的座標將會影響到傳輸範圍、訊號強度以及 Routing protocol 等，所以在建置無線網路劇本時，NSG2 會在畫面下方顯示出座標的資訊作為參考。並且 NSG 會假設無線網路 Node 的傳輸距離為 250m，所以當二個 Node 的傳輸距離在 250m 以內時，NSG2 會在二者之間畫一條連線，使用者可以透過滑鼠點選 Node 並拖曳來調整所要擺放的 Node 位置。

參、 通訊協定模式(Agent mode)

在 Agent mode 中，使用者可設定要附加在 Node 上的通訊協定（在 ns2 中稱為 Agent），目前 NSG2 主要支援 TCP（包含多種變形）和 UDP 通訊協定。在 Agent mode 除了設定 Agent 所屬的 Node 之外，還可以設定傳送端和接收端的相關資訊。

肆、 應用程式模式(Application mode)

在 Application mode 模式底下可以用來設定最上層的應用程式，例如 CBR 與 FTP，設定時只要在要附加的 Agent 上按一下滑鼠左鍵，然後再點選要擺放 Application 的位置即可，同樣的 Application 的重點在於附加的 Agent 而不是在於其位置，所以位置的設定只要在清楚容易辨識的位置即可。

伍、 參數設定(Parameters)

另外還有參數設定的功能，在建立無線網路劇本時，則另外多了無線網路（Wireless）及頻道特性（Channel）的相關設定。當一切都設定完畢之後，按下 TCL 按鈕即可產生 TCL 劇本檔，使用者這時候還可以自行在劇本檔上作一些調整，調整完畢後便可以將檔案儲存並利用 NS2 去執行。

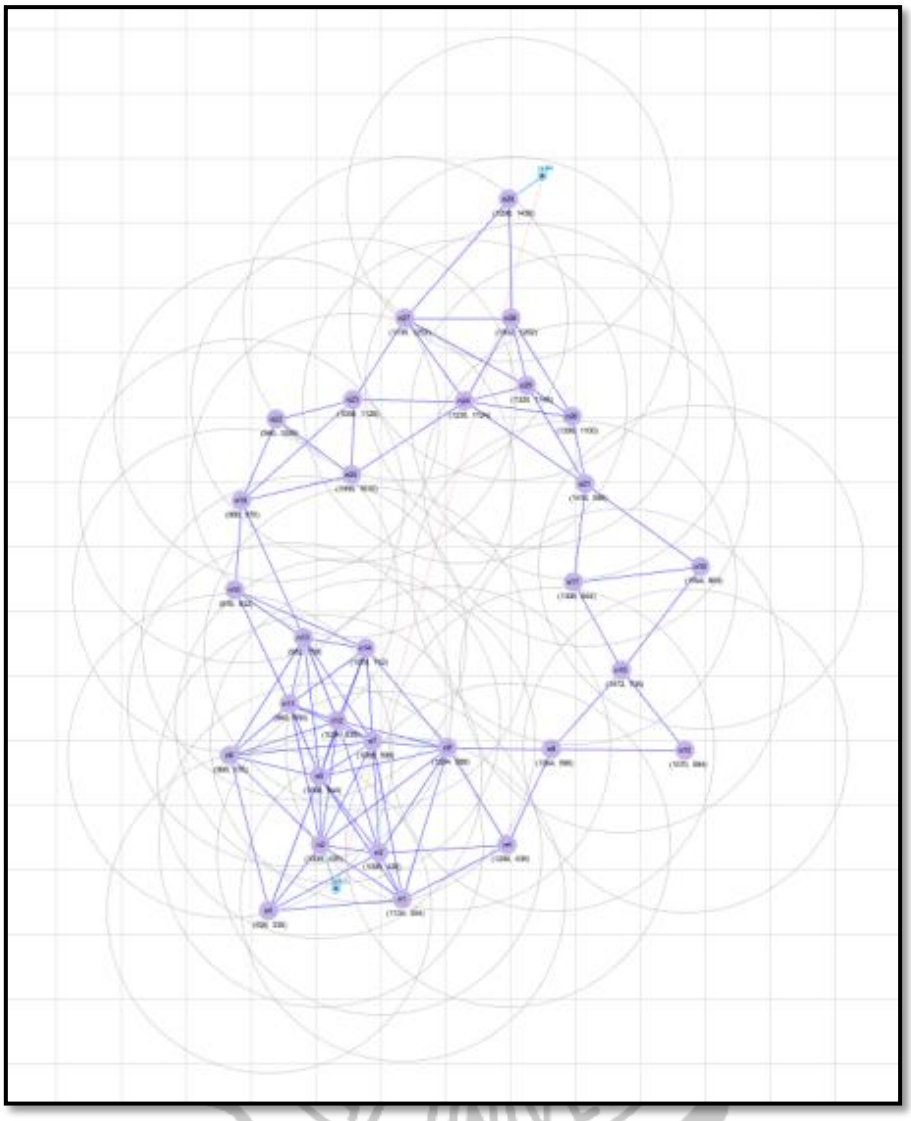


圖 4- 1 NSG2 劇本模擬

圖 12 即為運用 NSG2 劇本產生器所設計，並產生的 NS2 無線網路實驗模擬圖。再以產生的 TCL 劇本檔，進行調整與修改設計實驗。

第二節 網路效能測量

壹、 實驗目的

測量以 UDP 為傳輸協定的應用程式之封包遺失率（packet loss rate）、封包延遲（packet delay）、抖動率（packet jitter）、和吞吐量（throughput）。

貳、 背景知識

當模擬結束後，我們通常會需要使用模擬過程中所產生的模擬過程記錄檔來分析以得到封包遺失率、封包延遲、抖動率和吞吐量。所採用的方法是當有一條以 UDP 為傳輸協定的應用程式開始發送資料時，當封包從應用程式層到 UDP 層時，我們就把封包的序號、時間、和大小記錄到一個檔案中（發送端記錄檔），而當封包到達接收端時，同樣地我們把收到的封包序號、傳送時間（在傳送端要發送封包時會把當時的時間放在封包的標頭檔[common header]中）、到達時間、封包延遲時間（= 到達時間 - 傳送時間）、和封包大小記錄到另一個檔案中（接收端記錄檔），有了這兩個記錄檔，我們要求得吞吐量、封包延遲、抖動率、或封包遺失率就變得很簡單了。 模擬結束後就會產生 sd_udp 和 rd_udp 兩個記錄檔。

參、 awk 語言簡介

awk 是一種程式語言，具有一般程式語言常見的功能。awk 語言具有某些特點，如:使用直譯器(Interpreter)不需先行編譯；變數無型別之分(Typeless)，可使用文字當陣列的註標(Associative Array)等，因此，使用 awk 撰寫程式比起使用其它語言更簡潔便利且節省時間。另外，awk 還具有一些內建功能，使得 awk 擅於處理具資料列(Record)、欄位(Field)型態的資料；最後，awk 內建有 pipe 的功能，可將處理中的資料傳送給外部的 Shell 命令加以處理，再將 Shell 命令處理後的資料傳回 awk 程式，這個特點也使得 awk 程式很容易使用系統資源。

肆、 gnuplot 簡介

做完數值分析後，最重要的就是要把數值的結果畫成圖呈現出來，本實驗中所採用的繪圖工具是 gnuplot，命令導向的交談式繪圖程式(command-driven interactive function plotting program)。使用者輸入的每一項命令，可以逐步設定或修改繪圖環境。以圖形表達數據或函數，使我們可以藉由圖形做更進一步的分析。

伍、 封包遺失率（packet loss rate）

對於封包遺失率，我們可以先計算發送端記錄檔中有多少筆記錄，每一筆記錄就代表一個送出封包的資訊，因此有多少筆記錄就代表有多少封包被送出，同理，我們去計算接收端記錄檔中有多少記錄，每一筆記錄代表每一個所接收封包的資訊，因此有多少筆記錄就代表有多少封包被接收，從這兩個數值的差值我們就可以得知會有多少的封包在傳送的過程被丟棄了，然後再把這個差值除以全部送出的封包量，就可以得到封包遺失率。

計算 CBR 的封包遺失率：從 sd_udp 檔案中，可以得知共有幾筆封包傳送記錄；從 rd_udp 檔案中，可以得知共多少筆封包接收記錄，相減後的差值即為封包遺失的數量，因此封包遺失率為封包遺失數/傳送封包數。

陸、 封包延遲（packet delay）

封包延遲（packet delay）即為封包到達時間與封包傳送時間的差值。對於封包延遲，則可以直接從接收端記錄檔的第四欄得到。

求得封包延遲時間執行方法:

$awk ‘{print $1, $4}’ rd_udp > cbr_delay

使用 gnuplot 畫出 cbr_delay

```
gnuplot> plot “cbr_delay” title ‘cbr: packet delay’ with linespoints 1
gnuplot> set xlabel ‘packet sequence’
gnuplot> set ylabel ‘delay time (sec)’
gnuplot> set terminal gif
gnuplot> set output “cbr_delay.gif”
gnuplot> replot
```

柒、 抖動率(jitter)

抖動率就是延遲時間變化量(delay variance)，由於網路的流量隨時都在變化，

當流量大的時候，許多封包就必需在節點的佇列中等待被傳送，因此每個封包從傳送端到目的地端的時間也就不一定會相同，而這個不同的差異就是所謂的抖動率。抖動率越大，則表示網路越不穩定。

對於抖動率（jitter），則可以使用封包延遲時間差距除以封包序號差距得到（jitter ＝((recvtime(j) - sendtime(j)) - (recvtime(i) - sendtime(i))) / (j-i) = (delay(j) - delay(i)) /(j-i), 其中 j>i）。

計算抖動率：measure-jitter.awk

```
BEGIN{
	last_pkt_id   = -1;
	last_e2e_delay = -1;
}
{	pkt_id = $1;
	send_time = $2;
	rcv_time = $3;
	e2e_delay = $4;
	pkt_size = $5;
	if( last_pkt_id !=-1) {
		jitter = (e2e_delay - last_e2e_delay) / (pkt_id - last_pkt_id);
		printf("%f %f\n", send_time, jitter);
}
	last_pkt_id = pkt_id;
	last_e2e_delay = e2e_delay;
}
```

執行方法:

$awk –f measure-jitter.awk rd_udp > cbr_jitter

使用 gnuplot 畫出 cbr_jitter

```
gnuplot> plot “cbr_jitter” title ‘cbr: packet jitter’with linespoints 1
gnuplot> set xlabel ‘packet start time (sec)’
gnuplot> set ylabel ‘jitter (sec)’
gnuplot> set terminal gif
gnuplot> set output “cbr_jitter.gif”
gnuplot> replot
```

捌、 吞吐量（throughput）

吞吐量（throughput）為單位時間內，所能負荷的封包容量。對於吞吐量，則可以把所接收的封包大小總和除以所花費的時間就可以得到。

計算吞吐量:measure-throughput.pl

```
#使用方法: perl measure-throughput.pl <trace file> <granlarity>
#記錄檔檔名
$infile=$ARGV[0];
#多少時間計算一次(單位為秒)
$granularity=$ARGV[1];
$sum=0;
$sum_total=0;
$clock=0;
$maxrate=0;
$init=0;
#打開記錄檔
open (DATA,"<$infile")
|| die "Can't open $infile $!";
#讀取記錄檔中的每行資料,資料是以空白分成眾多欄位
while (<DATA>) {
    @x = split(' ');
     if($init==0){
     $start=$x[2];
     $init=1;
     }
#讀取的第零個欄位是 pkt_id
#讀取的第一個欄位是封包傳送時間
#讀取的第二個欄位是封包接收時間
#讀取的第三個欄位是封包 end to end delay
#讀取的第四個欄位是封包大小
    #判斷所讀到的時間,是否已經達到要統計吞吐量的時候
    if ($x[2]-$clock <= $granularity)
    {
        #計算單位時間內累積的封包大小
        $sum=$sum+$x[4];
```

```
        #計算累積的總封包大小
        $sum_total=$sum_total+$x[4];
    }
    else
    {
        #計算吞吐量
        $throughput=$sum*8.0/$granularity;
        if ($throughput > $maxrate){
            $maxrate=$throughput;
        }
        #輸出結果: 時間 吞吐量(bps)
        print STDOUT "$x[2]: $throughput bps\n";
        #設定下次要計算吞吐量的時間
        $clock=$clock+$granularity;
        $sum_total=$sum_total+$x[4];
        $sum=$x[4];
    }
}
$endtime=$x[2];
#計算最後一次的吞吐量大小
$throughput=$sum*8.0/$granularity;
print STDOUT "$x[2]: $throughput bps\n";
$clock=$clock+$granularity;
$sum=0;
#print STDOUT "$sum_total $start $endtime\n";
$avgrate=$sum_total*8.0/($endtime-$start);
print STDOUT "Average rate: $avgrate bps\n";
print STDOUT "Peak rate: $maxrate bps\n";
#關閉檔案
close DATA;
exit(0);
```

執行的方法:

$perl measure-throughput.pl rd_udp 1

執行的結果

Average rate: bps

第三節 實驗參數

實驗初始設定為設置 150 個節點，並預設一個空隙範圍。實驗於第五秒開始傳送封包，於第 100 秒停止封包傳送，而模擬時間於第 120 秒時結束實驗，節點的傳輸範圍為 250m。初始實驗設定參數為佇列預設為 50，封包大小為 512 bytes，而傳輸量為 0.2 Mb。分別模擬在通訊空隙環境下 GGVR、GAR 和 AODV 的路由表現狀況，並分別在佇列、封包大小和傳輸量控制變因的調整下，進行在不同環境下網路效能的測量與比較。

表 4- 1 實驗初始設定

Parameter Setting	
Node	150
Start time	5 sec
Stop time	100 sec
Simulation time	120 sec
Transmission Range	250m
佇列	50
封包大小	512 bytes
傳輸量	0.2 Mb

第四節 控制變因- 佇列

壹、 封包遺失率（packet loss rate）

表 4- 2 佇列封包遺失率

Queue	5	10	15	25	50	75
GGVR	0.5327	0.4912	0.5025	0.456	0.4843	0.4608
GAR	0.6698	0.6318	0.5719	0.604	0.6436	0.6333
AODV	0.4112	0.4482	0.5504	0.570	0.5354	0.4971

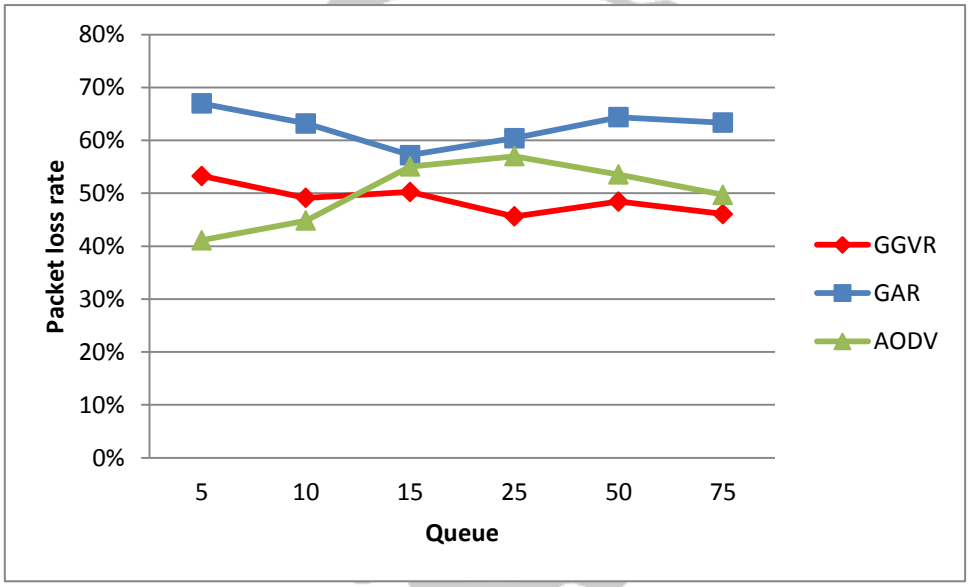


圖 4- 2 佇列封包遺失率

在控制變因為佇列時，封包遺失率在佇列為 5 時，AODV 的封包遺失率最低，GGVR 次之，而 GAR 的封包遺失率最高。封包遺失率在佇列為 10 時，AODV 的封包遺失率最低，GGVR 次之但差距甚小，而 GAR 的封包遺失率最高，且高出甚多。封包遺失率在佇列為 15 時，GGVR 的封包遺失率最低，AODV 次之，而 GAR 的封包遺失率最高，但三者差距甚小，封包遺失率狀況差異不明顯。封包遺失率在佇列為 25 時，GGVR 的封包遺失率最低，且有顯著差距。而 AODV 次之，GAR 的封包遺失率最高，但兩者差距甚小。封包遺失率在佇列為 50 時，GGVR

的封包遺失率最低，AODV 次之，而 GAR 的封包遺失率最高。封包遺失率在佇列為 75 時，GGVR 的封包遺失率最低，AODV 次之，但兩者差距甚小。而 GAR 的封包遺失率最高，且有顯著差距。GGVR 的封包遺失率在佇列為 15、25、50 和 75 時，遺失率較低。尤其在佇列為 25 時表現最佳，可能是有效迴避空隙的路由方式，在此時發揮了效果。

貳、 封包延遲（packet delay）

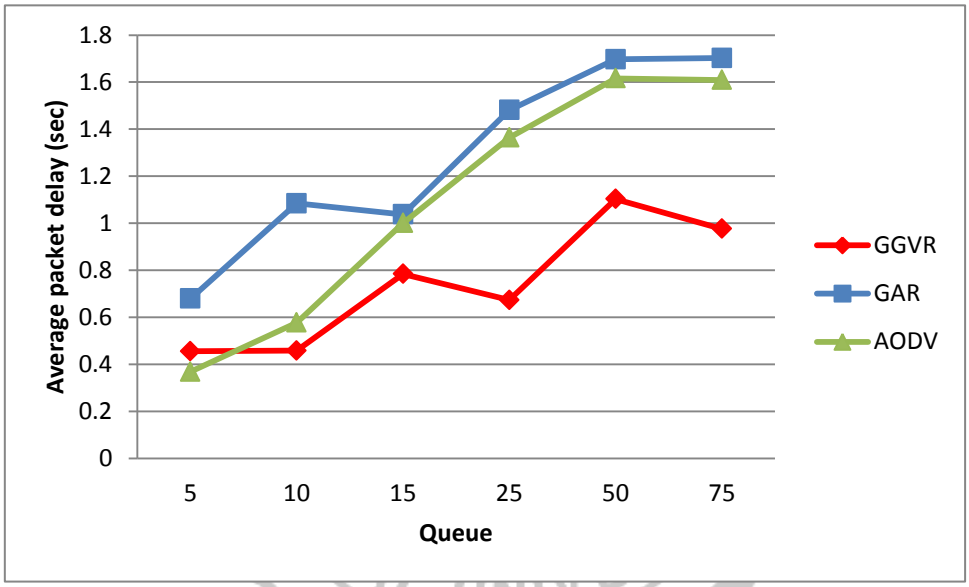


圖 4- 3 佇列平均封包延遲時間

在控制變因為佇列時，平均封包延遲時間在佇列為 5 時，AODV 的平均封包延遲時間最少，GGVR 次之，而 GAR 的平均封包延遲時間最多。平均封包延遲時間在佇列為 10 時，GGVR 的平均封包延遲時間最少，AODV 次之，但差距甚小。而 GAR 的平均封包延遲時間最多，且高出甚多。平均封包延遲時間在佇列為 15 時，GGVR 的平均封包延遲時間最少，且有些為差距。而 AODV 次之，GAR 的平均封包延遲時間最多，但兩者差距甚小。平均封包延遲時間在佇列為 25 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的平均封包延遲時間最多，但兩者差距甚小。平均封包延遲時間在佇列為 50 時，GGVR

的平均封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的平均封包延遲時間最多。平均封包延遲時間在佇列為 75 時，GGVR 的平均封包延遲時間最少，AODV 次之，但兩者差距甚小。而 GAR 的平均封包延遲時間最多，且有顯著差距。GGVR 的平均封包延遲時間在佇列為 10、15、25、50 和 75 時，平均封包延遲時間較少。尤其在佇列為 15、25 與 50 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

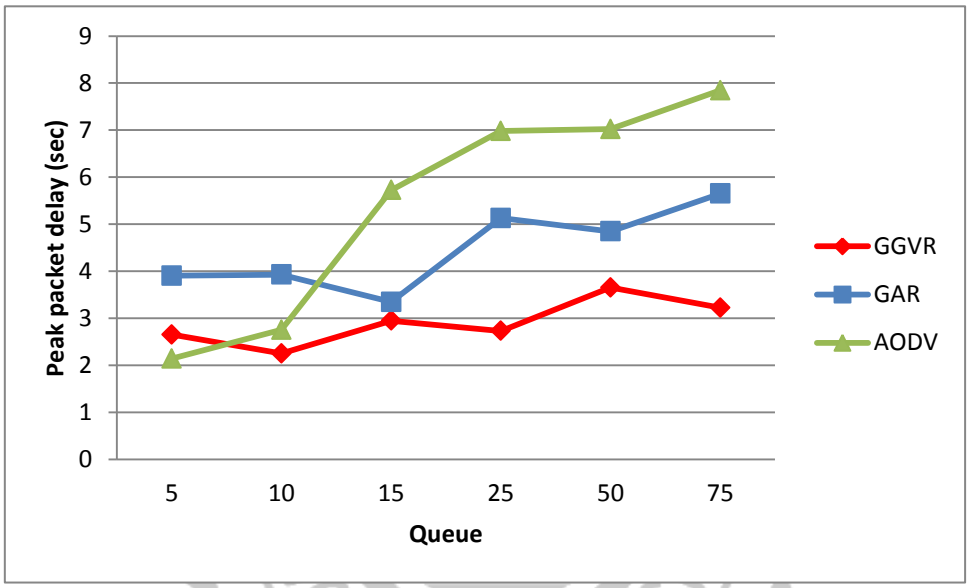


圖 4- 4 佇列最高封包延遲時間

在控制變因為佇列時，最高封包延遲時間在佇列為 5 時，AODV 的最高封包延遲時間最少，GGVR 次之，而 GAR 的最高封包延遲時間最多，但三者差距不大。最高封包延遲時間在佇列為 10 時，GGVR 的最高封包延遲時間最少，AODV 次之，但差距甚小。而 GAR 的最高封包延遲時間最多，有些微差距。最高封包延遲時間在佇列為 15 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者差距甚小。而 AODV 的最高封包延遲時間最多，且高出甚多。最高封包延遲時間在佇列為 25 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者有顯著差距。而 AODV 的最高封包延遲時間最多，差距甚多。最高封包延遲時間在佇列為 50 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者差距甚小。而 AODV 的最高封包延遲時間最多，有些微差距。最高封包延遲時間在佇列為 75 時，GGVR

的最高封包延遲時間最少，GAR 次之，兩者有顯著差距。而 AODV 的最高封包延遲時間最多，且有顯著差距。GGVR 的最高封包延遲時間在佇列為 10、15、25、50 和 75 時，最高封包延遲時間較少。尤其在佇列為 25、50 與 75 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

參、 抖動率（packet jitter）

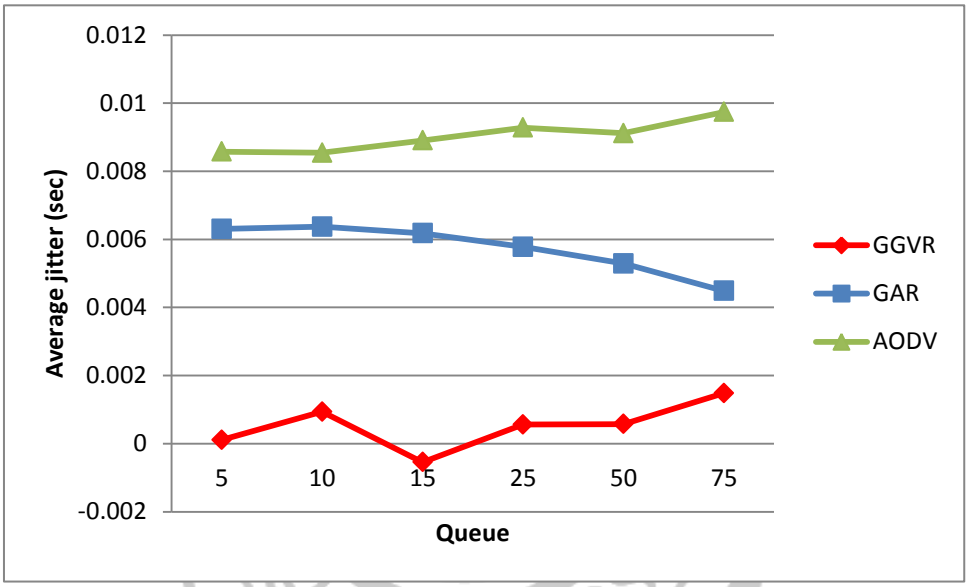


圖 4- 5 佇列平均抖動率

在控制變因為佇列時，平均抖動率在佇列為 5 時，GGVR 的平均抖動率最少，且有顯著差距。而 AODV 次之，GAR 的平均抖動率最多，兩者有些微差距。平均抖動率在佇列為 10 時，GGVR 的平均抖動率時間最少，且有顯著差距。而 GAR 次之，AODV 的平均抖動率最多，兩者有些微差距。平均抖動率在佇列為 15 時，GGVR 的平均抖動率最少，且有顯著差距。而 GAR 次之，AODV 的平均抖動率最多，兩者有些微差距。平均抖動率在佇列為 25 時，GGVR 的平均抖動率最少，且有顯著差距。而 GAR 次之，AODV 的平均抖動率最多，兩者有些微差距。平均抖動率在佇列為 50 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，有顯著差距。平均抖動率在佇列為 75 時，GGVR

的平均抖動率最少，GAR 次之，兩者有顯著差距。而 AODV 的平均抖動率最多，有些微差距。GGVR 的平均封包延遲時間在佇列的變化下，平均抖動率較少，表現狀況皆顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

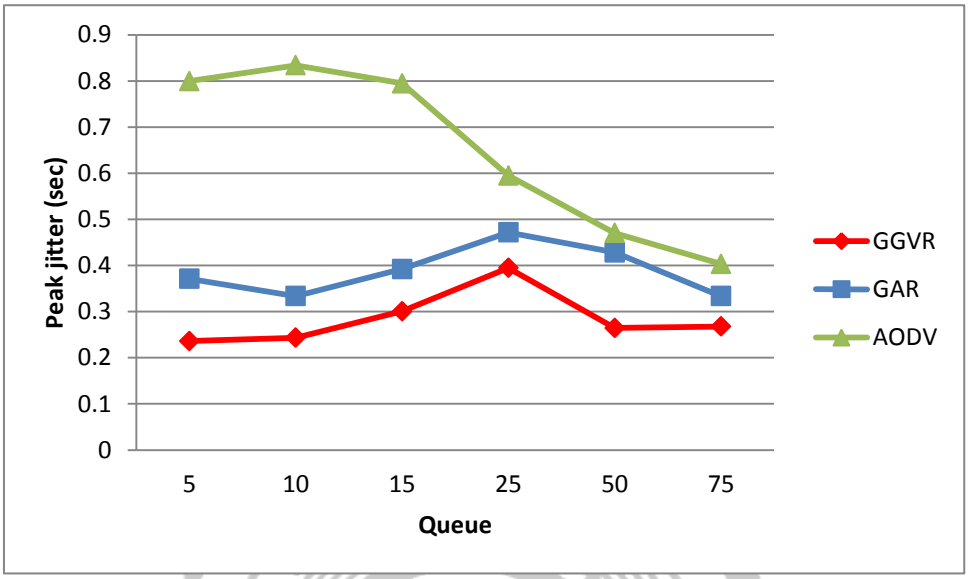


圖 4- 6 佇列最高抖動率

在控制變因為佇列時，最高抖動率在佇列為 5 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，且有顯著差距。最高抖動率在佇列為 10 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，有顯著差距。最高抖動率在佇列為 15 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，且有顯著差距。最高抖動率在佇列為 25 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，且有顯著差距。最高抖動率在佇列為 50 時，GGVR 的最高抖動率最少，且有顯著差距。而 GAR 次之，AODV 的最高抖動率最多，兩者差距不大。最高抖動率在佇列為 75 時，GGVR 的最高抖動率最少，GAR 次之，而 AODV 的最高抖動率最多，但三者差距不大。GGVR 的最高抖動率在佇列為 5、10、15、25 和 50 時，最高抖動率較少。尤其在佇列為 50 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

肆、　吞吐量（throughput）

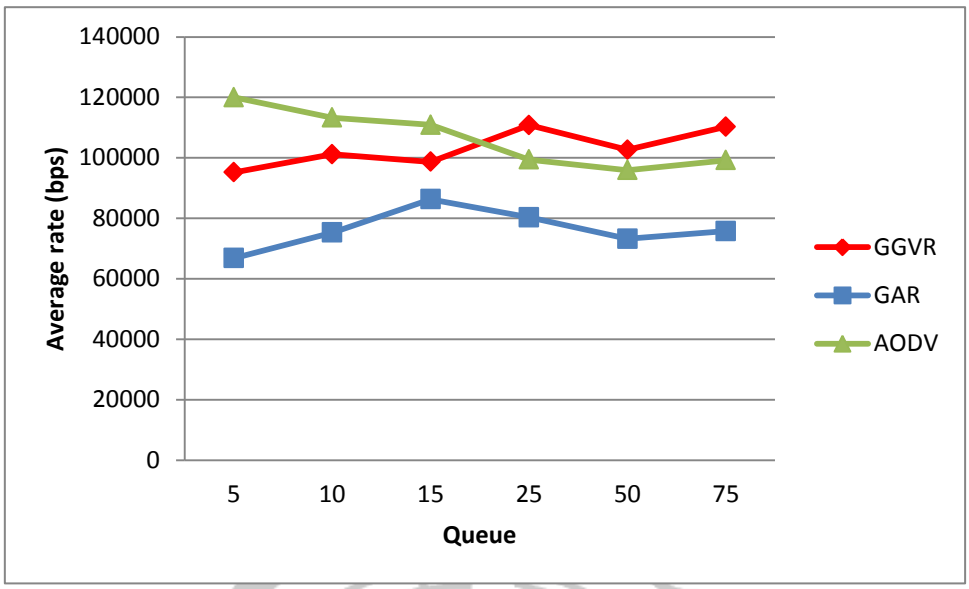


圖 4-7 佇列平均吞吐量

在控制變因為佇列時，平均吞吐量在佇列為 5 時，GAR 的平均吞吐量最少，GGVR 次之，而 AODV 的平均吞吐量最多，三者皆有顯著差距。平均吞吐量在佇列為 10 時，GAR 的平均吞吐量最少，且有顯著差距。而 GGVR 次之，AODV 的平均吞吐量最多，但兩者差距不大。平均吞吐量在佇列為 15 時，GAR 的平均吞吐量最少，GGVR 次之，而 AODV 的平均吞吐量最多，但三者差距甚小。平均吞吐量在佇列為 25 時，GAR 的平均吞吐量最少，AODV 次之，兩者有些微差距。而 GGVR 的平均吞吐量最多，且差距甚多。平均吞吐量在佇列為 50 時，GAR 的平均吞吐量最少，AODV 次之，而 GGVR 的平均吞吐量最多，三者皆有些微差距。平均吞吐量在佇列為 75 時，GAR 的平均吞吐量最少，且有顯著差距。而 AODV 次之，GGVR 的平均吞吐量最多，但兩者差距甚小。GGVR 的平均吞吐量在佇列為 25、50 和 75 時，平均吞吐量較高。尤其在佇列為 25 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

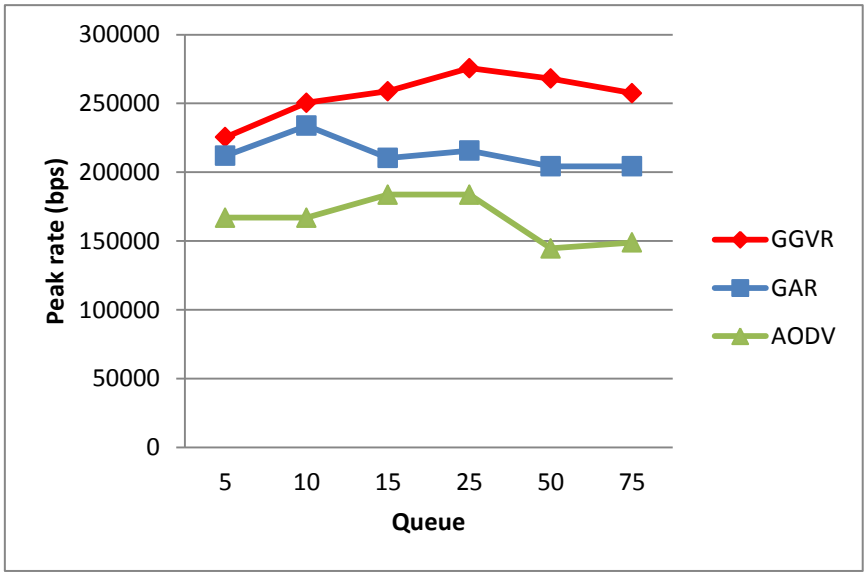


圖 4- 8 佇列最高吞吐量

在控制變因為佇列時，最高吞吐量在佇列為 5 時，AODV 的最高吞吐量最少，GAR 次之，兩者有些微差距。而 GGVR 的最高吞吐量最多，且有顯著差距。最高吞吐量在佇列為 10 時，AODV 的最高吞吐量最少，且有顯著差距。而 GAR 次之，GGVR 的最高吞吐量最多，但兩者差距不大。最高吞吐量在佇列為 15 時，AODV 的最高吞吐量最少，GAR 次之，兩者有些微差距。而 GGVR 的最高吞吐量最多，且有顯著差距。最高吞吐量在佇列為 25 時，AODV 的最高吞吐量最少，且有顯著差距。而 GAR 次之， GGVR 的最高吞吐量最多，兩者有些微差距。最高吞吐量在佇列為 50 時，AODV 的最高吞吐量最少，GAR 次之，而 GGVR 的最高吞吐量最多，三者皆有些顯著差距。最高吞吐量在佇列為 75 時，AODV 的最高吞吐量最少，GAR 次之，兩者差距甚小。而 GGVR 的最高吞吐量最多，且有些微差距。GGVR 的最高吞吐量在佇列為 5、10、15、25、50 和 75 時，最高吞吐量較高。尤其在佇列為 15、25、50 和 75 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

第五節 控制變因- 封包大小

壹、 封包遺失率（packet loss rate）

表 4- 3 封包大小封包遺失率

Packet size	128 bytes	256 bytes	512 bytes	1024 bytes
GGVR	0.7845	0.5906	0.5943	0.4025
GAR	0.8122	0.6860	0.6436	0.6017
AODV	0.7307	0.6162	0.5354	0.4849

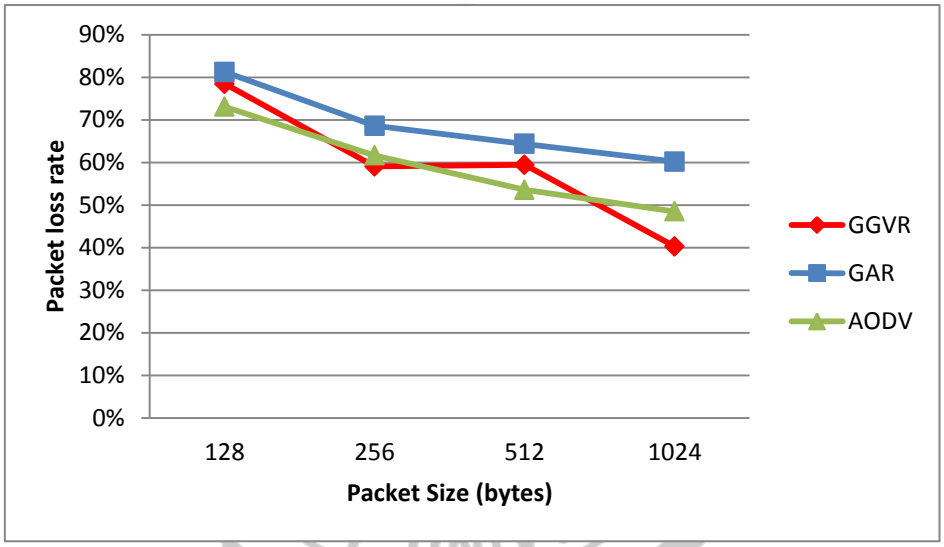


圖 4- 9 封包大小封包遺失率

在控制變因為封包大小時，封包遺失率在封包大小為 128 bytes 時，AODV 的封包遺失率最低，GGVR 次之，而 GAR 的封包遺失率最高。封包遺失率在封包大小為 256 bytes 時，GGVR 的封包遺失率最低，AODV 次之，但兩者差距甚小，而 GAR 的封包遺失率最高。封包遺失率在封包大小為 512 bytes 時，AODV 的封包遺失率最低， GGVR 次之，而 GAR 的封包遺失率最高。封包遺失率在封包大小為 1024 bytes 時，GGVR 的封包遺失率最低，AODV 次之，而 GAR 的封包遺失率最高，且有顯著差距。GGVR 的封包遺失率在封包大小為 256 bytes 與 1024 bytes 時，封包遺失率較低。而在封包大小為 1024 bytes 時，有顯著差距，可能是有效避免空隙的路由方式，在此時發揮了效果。

貳、 封包延遲（packet delay）

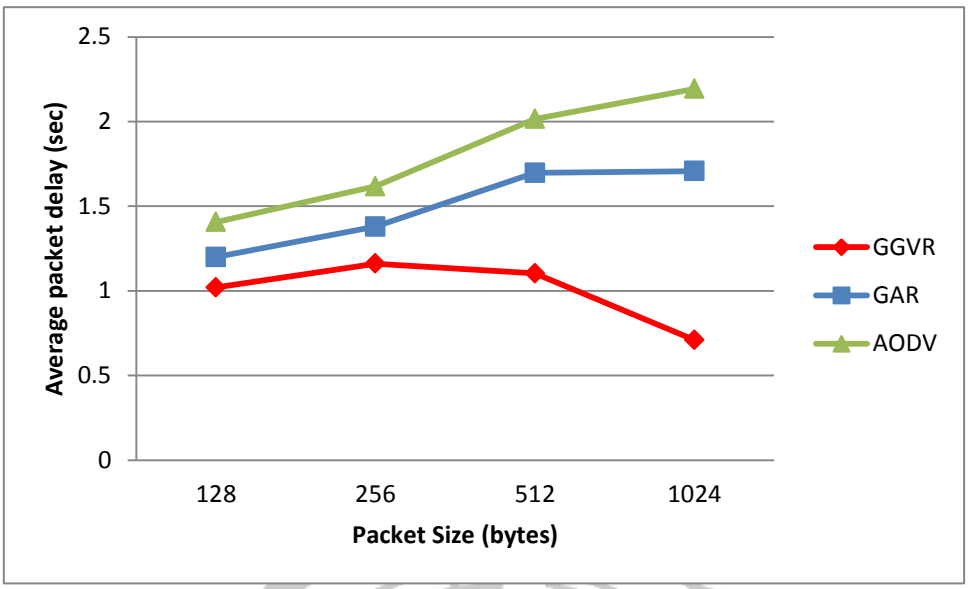


圖 4- 10 封包大小平均封包延遲時間

在控制變因為封包大小時，平均封包延遲時間在封包大小為 128 bytes 時，GGVR、GAR 和 AODV 三者的平均封包延遲時間皆不高，皆有良好表現。平均封包延遲時間在封包大小為 256 bytes 時，GGVR 的平均封包延遲時間最少，GAR 次之，而 AODV 的平均封包延遲時間最多，皆有些微差距。平均封包延遲時間在封包大小為 512 bytes 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，且有些微差距。平均封包延遲時間在封包大小為 1024 bytes 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，且有顯著差距。尤其在封包大小為 256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

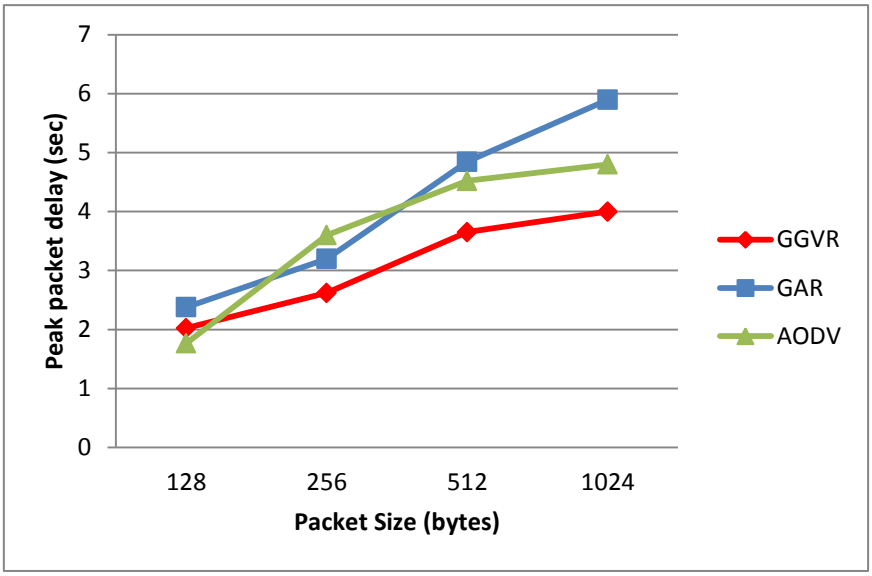


圖 4- 11 封包大小最高封包延遲時間

在控制變因為封包大小時，最高封包延遲時間在封包大小為 128 bytes 時，GGVR、GAR 和 AODV 三者的最高封包延遲時間皆不高，皆有良好表現。最高封包延遲時間在封包大小為 256 bytes 時，GGVR 的最高封包延遲時間最少，GAR 次之，而 AODV 的最高封包延遲時間最多，皆有些微差距。最高封包延遲時間在封包大小為 512 bytes 時，GGVR 的最高封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的最高封包延遲時間最多，且差距甚大。最高封包延遲時間在封包大小為 1024 bytes 時，GGVR 的最高封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的最高封包延遲時間最多，且有顯著差距。GGVR 的最高封包延遲時間在封包大小的變化下，在封包大小為 256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

參、　抖動率（packet jitter）

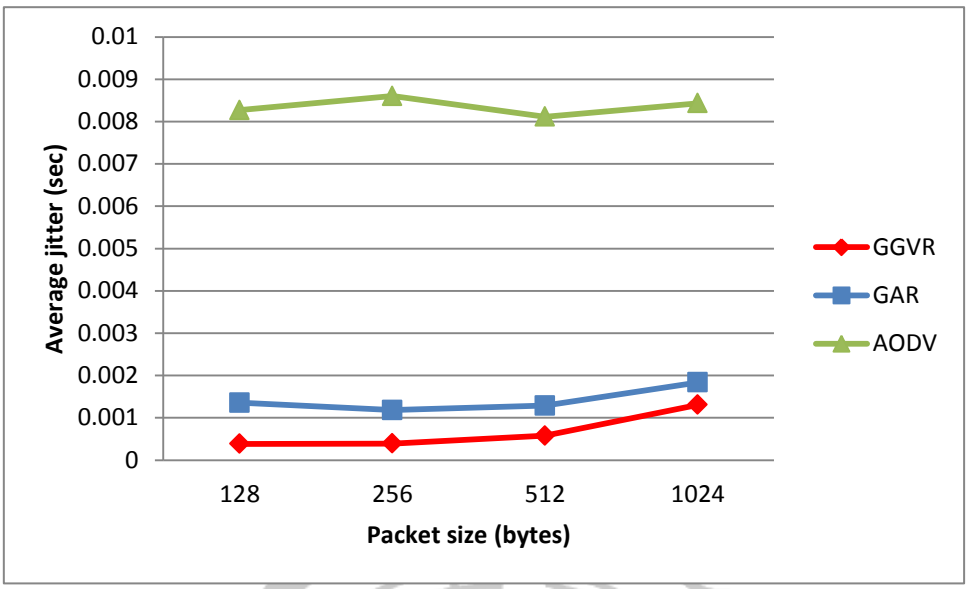


圖 4- 12 封包大小平均抖動率

在控制變因為封包大小時，平均抖動率在封包大小為 128 bytes 時， GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。平均抖動率在封包大小為 256 bytes 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。平均抖動率在封包大小為 512 bytes 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。平均抖動率在封包大小為 1024 bytes 時，GGVR 的平均抖動率最少，GAR 次之，兩者有顯著差距。而 AODV 的平均抖動率最多，有些微差距。GGVR 的平均抖動率在封包大小的變化下，平均抖動率皆較少，表現良好。尤其在封包大小為 128 bytes、256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

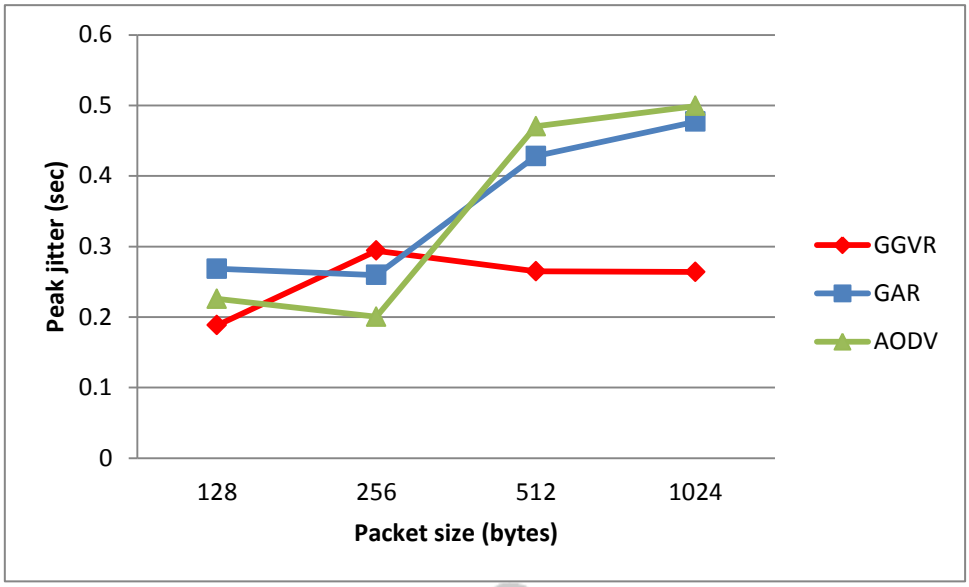


圖 4- 13 封包大小最高抖動率

在控制變因為封包大小時，最高抖動率在封包大小為 128 bytes 時，GGVR 的最高抖動率最少，AODV 次之，但兩者差距不大。而 GAR 的最高抖動率最多，有些微差距。最高抖動率在封包大小為 256 bytes 時，AODV 的最高抖動率最少，GAR 次之，而 GGVR 的最高抖動率最多，三者皆有些微差距。最高抖動率在封包大小為 512 bytes 時，GGVR 的最高抖動率最少，且有顯著差距。而 GAR 次之，AODV 的最高抖動率最多，兩者有些微差距。最高抖動率在封包大小為 1024 bytes 時，GGVR 的最高抖動率最少，GAR 次之，而 AODV 的最高抖動率最多，三者皆有顯著差距。GGVR 的最高封包延遲時間在封包大小的變化下，在封包大小為 128 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

肆、　吞吐量（throughput）

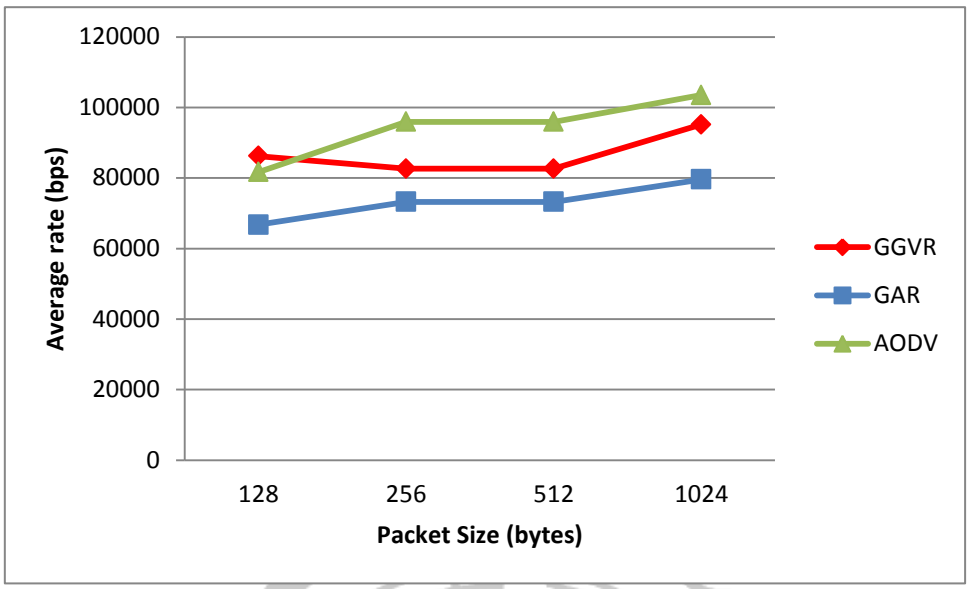


圖 4- 14 封包大小平均吞吐量

在控制變因為封包大小時，平均吞吐量在封包大小為 128 bytes 時，GAR 的平均吞吐量最少，且有顯著差距。而 AODV 次之，GGVR 的平均吞吐量最多，兩者有些微差距。平均吞吐量在封包大小為 256 bytes 時，GAR 的平均吞吐量最少，GGVR 次之，而 AODV 的平均吞吐量最多，三者皆有些微差距。平均吞吐量在封包大小為 512 bytes 時，GAR 的平均吞吐量最少， GGVR 次之，而 AODV 的平均吞吐量最多，三者皆有些微差距。平均吞吐量在封包大小為 1024 bytes 時，GAR 的平均吞吐量最少，且有顯著差距。而 GGVR 次之，AODV 的平均吞吐量最多，兩者差距不大。GGVR 的平均吞吐量在封包大小的變化下，在封包大小為 128 bytes 時表現狀況良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

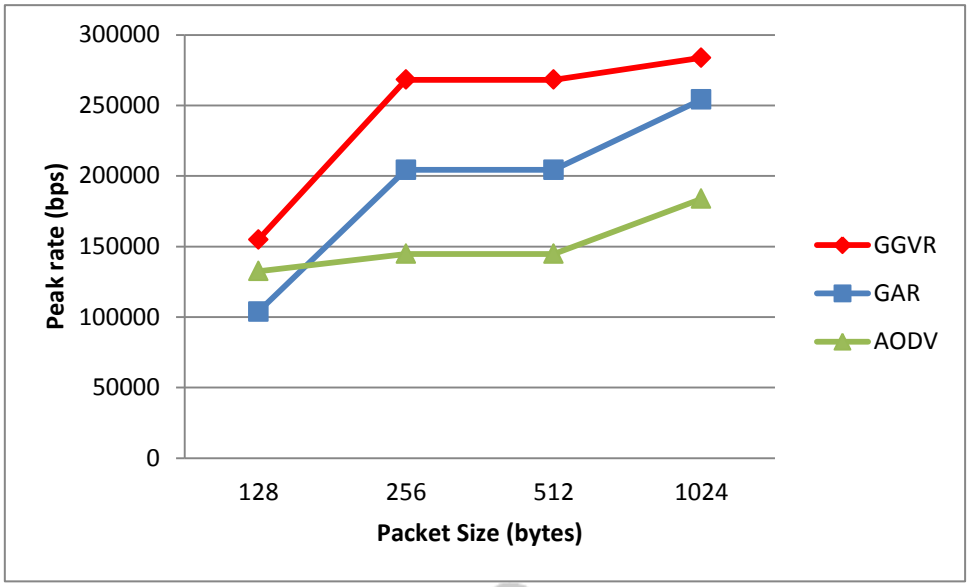


圖 4- 15 封包大小最高吞吐量

在控制變因為封包大小時，最高吞吐量在封包大小為 128 bytes 時，GAR 的最高吞吐量最少，AODV 次之，而 GGVR 的最高吞吐量最多，三者差距不大。最高吞吐量在封包大小為 256 bytes 時，AODV 的最高吞吐量最少，GAR 次之，而 GGVR 的最高吞吐量最多，三者皆有顯著差距。最高吞吐量在封包大小為 512 bytes 時，AODV 的最高吞吐量最少，GAR 次之，而 GGVR 的最高吞吐量最多，三者皆有顯著差距。最高吞吐量在封包大小為 1024 bytes 時，AODV 的最高吞吐量最少，GGVR 次之，兩者差距不大。而 GAR 的最高吞吐量最多，且有顯著差距。GGVR 的最高吞吐量在封包大小的變化下，在封包大小為 128 bytes、256 bytes、512 bytes 和 1024bytes 時，表現狀況良好。尤其在封包大小為 256 bytes 和 512 bytes 時，表現顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

第六節 控制變因- 傳輸量

壹、 封包遺失率（packet loss rate）

表 4- 4 傳輸量封包遺失率

Rate	0.2 mb	0.4 mb	0.6 mb	0.8 mb	1.0 mb
GGVR	0.5943	0.7311	0.8356	0.8728	0.8925
GAR	0.6436	0.8264	0.8741	0.9135	0.9330
AODV	0.5354	0.7322	0.8394	0.8914	0.9136

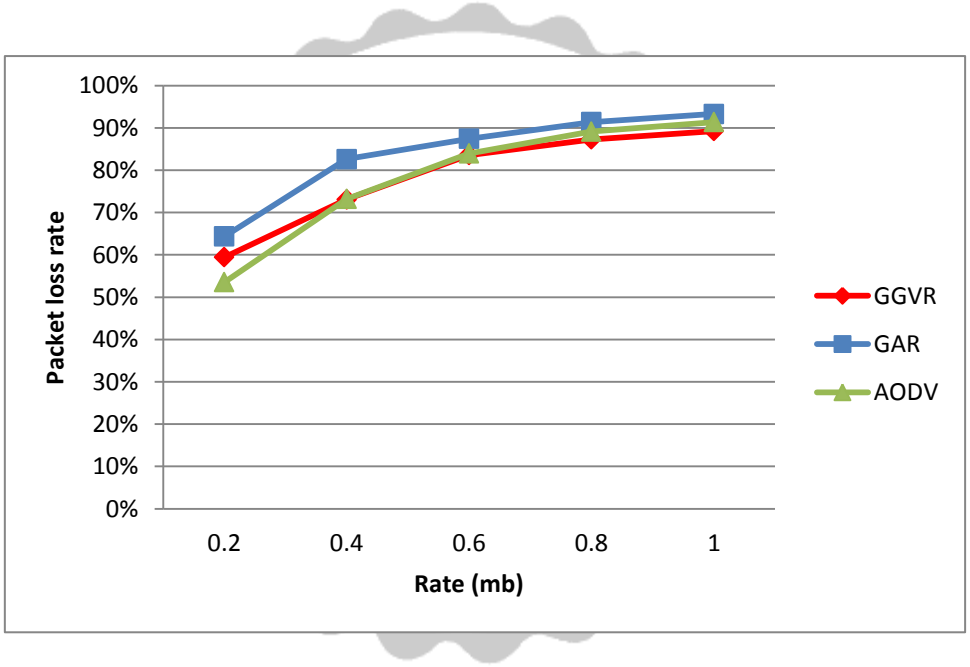


圖 4- 16 傳輸量封包遺失率

在控制變因為傳輸量時，封包遺失率在傳輸量為 0.2mb 時，AODV 的封包遺失率最低，GGVR 次之，而 GAR 的封包遺失率最高，但三者差距甚小。封包遺失率在封包大小為 0.4mb 時，GGVR 的封包遺失率最低，AODV 次之，但兩者差距甚小，而 GAR 的封包遺失率最高。封包遺失率在傳輸量為 0.6mb 時，AODV 的封包遺失率最低， GGVR 次之，而 GAR 的封包遺失率最高，但三者差距甚小。封包遺失率在封包大小為 0.8mb 時，GGVR 的封包遺失率最低，AODV 次之，而 GAR 的封包遺失率最高，但三者差距甚小。封包遺失率在傳輸量為 1.0mb 時，

GGVR 的封包遺失率最低，AODV 次之，而 GAR 的封包遺失率最高，但三者差距甚小。在控制變因為傳輸量時，GGVR 的封包遺失率在傳輸量為 0.4、0.6、0.8 和 1.0 時，封包遺失率表現狀況良好，封包遺失率較低，在傳輸量的變化下，皆能有穩定，且良好的表現。

貳、 封包延遲（packet delay）

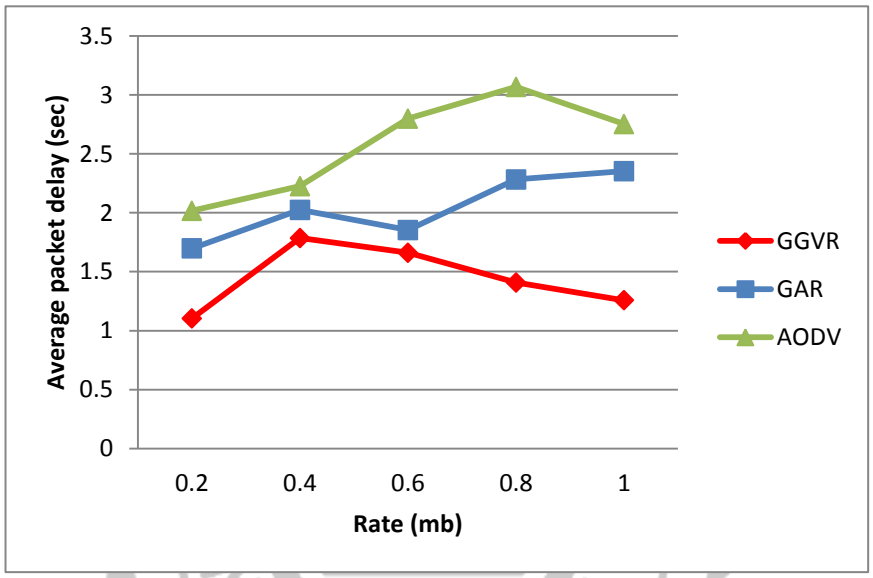


圖 4- 17 傳輸量平均封包延遲時間

在控制變因為傳輸量時，平均封包延遲時間在傳輸量為 0.2 mb 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，兩者差距甚小。平均封包延遲時間在傳輸量為 0.4 mb 時，GGVR 的平均封包延遲時間最少，GAR 次之，而 AODV 的平均封包延遲時間最多，皆有些微差距。平均封包延遲時間在傳輸量為 0.6 mb 時，GGVR 的平均封包延遲時間最少，GAR 次之，兩者有些微差距。而 AODV 的平均封包延遲時間最多，且有顯著差距。平均封包延遲時間在傳輸量為 0.8 mb 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，且有顯著差距。平均封包延遲時間在傳輸量為 1.0 mb 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，兩者有些微差

距。GGVR 的平均封包延遲時間在傳輸量的變化下，平均封包延遲時間皆較少，表現良好。尤其在傳輸量為 0.2 mb、0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

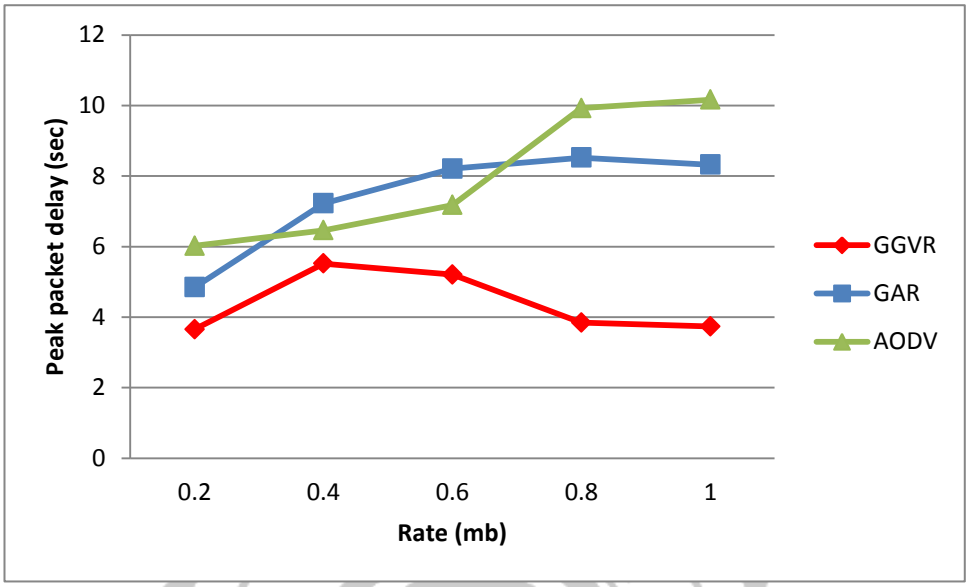


圖 4- 18 傳輸量最高封包延遲時間

在控制變因為傳輸量時，最高封包延遲時間在傳輸量為 0.2 mb 時，GGVR 的最高封包延遲時間最少，GAR 次之，而 AODV 的最高封包延遲時間最多，三者皆有顯著差距。最高封包延遲時間在傳輸量為 0.4 mb 時，GGVR 的最高封包延遲時間最少，AODV 次之，而 GAR 的最高封包延遲時間最多，但三者差距不大。最高封包延遲時間在傳輸量為 0.6 mb 時，GGVR 的最高封包延遲時間最少，有顯著差距。而 AODV 次之，GAR 的最高封包延遲時間最多，兩者有些微差距。最高封包延遲時間在傳輸量為 0.8 mb 時，GGVR 的最高封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的最高封包延遲時間最多，且有顯著差距。最高封包延遲時間在傳輸量為 1.0 最高時，GGVR 的最高封包延遲時間最少，GAR 次之，而 AODV 的最高封包延遲時間最多，三者皆有顯著差距。GGVR 的最高封包延遲時間在傳輸量的變化下，在傳輸量為 0.2 mb、0.4 mb、0.6 mb、0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

參、　抖動率（packet jitter）

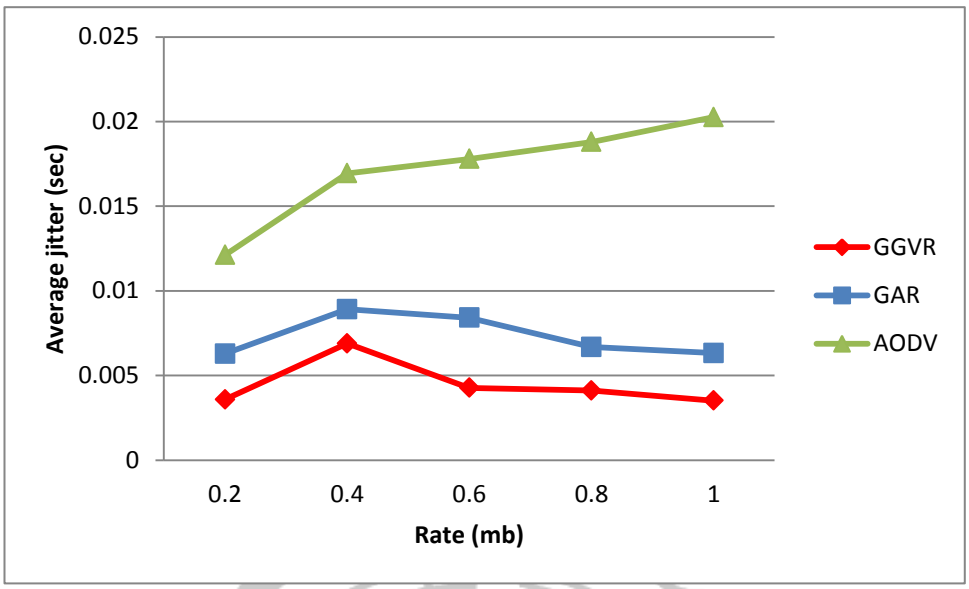


圖 4- 19 傳輸量平均抖動率

在控制變因為傳輸量時，平均抖動率在傳輸量為 0.2 mb 時，GGVR 的平均抖動率最少。而 GAR 次之，AODV 的平均抖動率最多，兩者差距顯著。平均抖動率在傳輸量為 0.4 mb 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。平均抖動率在傳輸量為 0.6 mb 時，GGVR 的平均抖動率最少，GAR 次之，而 AODV 的平均抖動率最多，三者皆有顯著差距。平均抖動率在傳輸量為 0.8 mb 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。平均抖動率在傳輸量為 1.0 mb 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。GGVR 的平均抖動率在傳輸量的變化下，平均抖動率皆較少，表現良好。尤其在傳輸量為 0.4 mb、0.6 mb、0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

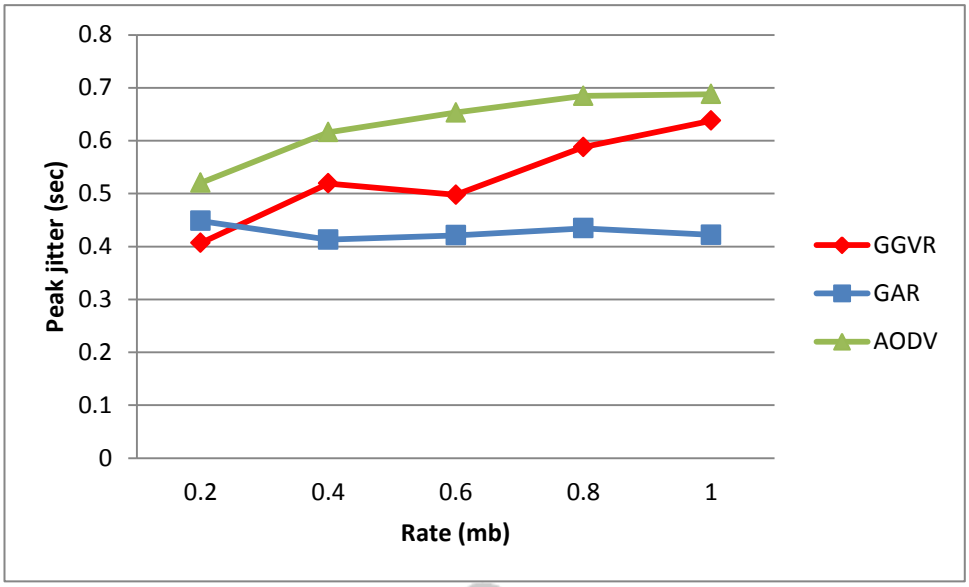


圖 4- 20 傳輸量最高抖動率

在控制變因為傳輸量時，最高抖動率在傳輸量為 0.2 mb 時，GGVR 的最高抖動率最少，有些微差距。而 GAR 次之，AODV 的最高抖動率最多，兩者只有些微差距。最高抖動率在傳輸量為 0.4 mb 時，GAR 的最高抖動率最少，且有顯著差距。而 GGVR 次之，AODV 的最高抖動率最多，兩者有些微差距。最高抖動率在傳輸量為 0.6 mb 時，GAR 的最高抖動率最少，GGVR 次之，而 AODV 的最高封包延遲時間最多，三者有顯著差距。最高抖動率在傳輸量為 0.8 mb 時，GAR 的最高抖動率最少，且有顯著差距。而 GGVR 次之，AODV 的最高抖動率最多，且有顯著差距。最高抖動率在傳輸量為 1.0 最高時，GAR 的最高封包延遲時間最少，且有顯著差距。而 GGVR 次之，AODV 的最高抖動率最多，兩者差距不大。GGVR 的最高封包延遲時間在傳輸量的變化下，在傳輸量為 0.2 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

肆、　吞吐量（throughput）

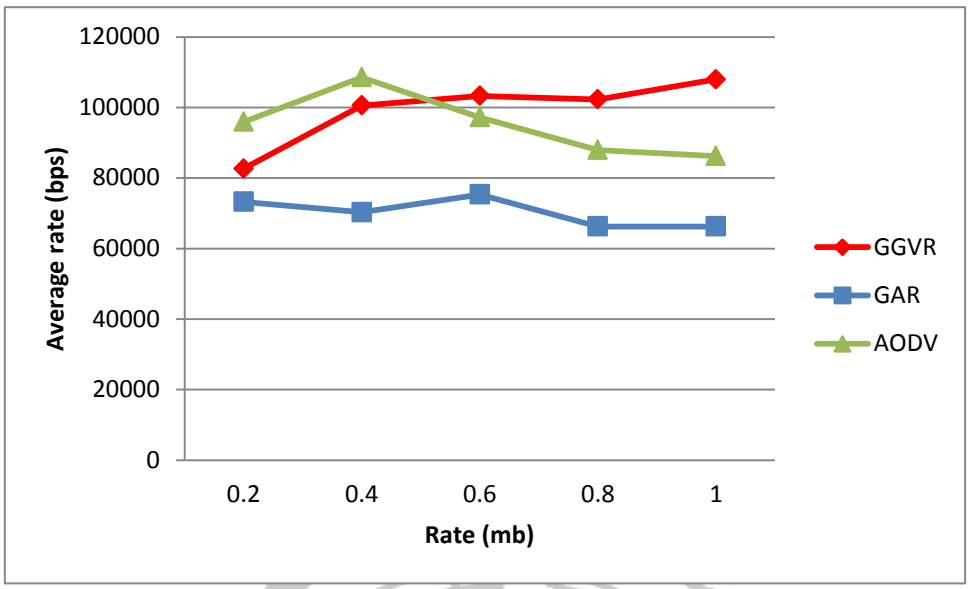


圖 4- 21 傳輸量平均吞吐量

在控制變因為傳輸量時，平均吞吐量在傳輸量為 0.2 mb 時，GAR 的平均吞吐量最少，GGVR 次之，而 AODV 的平均吞吐量最多，三者皆有些微差距。平均吞吐量在傳輸量為 0.4 mb 時，GAR 的平均吞吐量最少，且有顯著差距。而 GGVR 次之，AODV 的平均吞吐量最多，兩者差距不大。平均吞吐量在傳輸量為 0.6 mb 時，GAR 的平均吞吐量最少，且有顯著差距。而 AODV 次之，GGVR 的平均吞吐量最多，兩者差距不大。平均吞吐量在傳輸量為 0.8 mb 時，GAR 的平均吞吐量最少，AODV 次之，而 GGVR 的平均吞吐量最多，三者皆有顯著差距。平均吞吐量在傳輸量為 1.0 mb 時，GAR 的平均吞吐量最少，AODV 次之，而 GGVR 的平均吞吐量最多，三者皆有顯著差距。GGVR 的平均吞吐量在傳輸量的變化下，在傳輸量為 0.6 mb、0.8 mb 和 1.0 mb 時表現良好。尤其在傳輸量為 0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

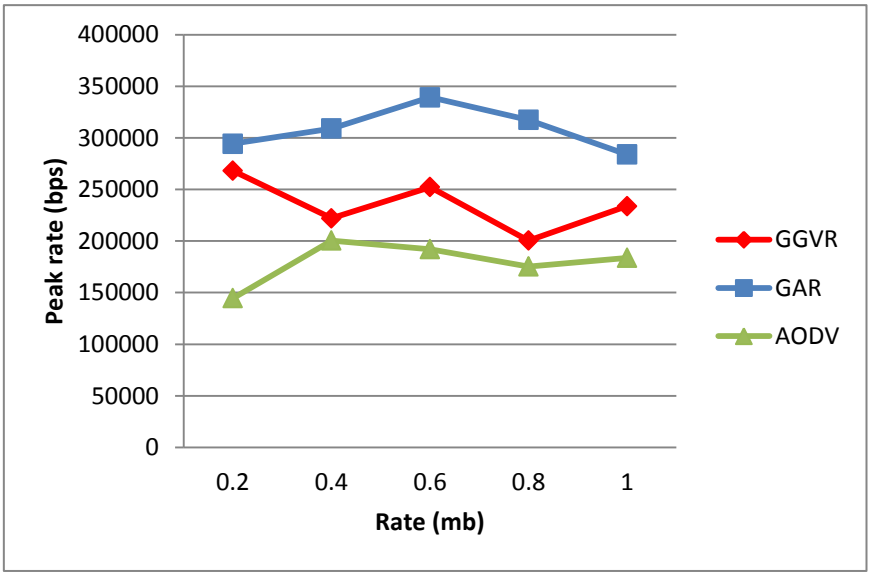


圖 4- 22 傳輸量最高吞吐量

在控制變因為傳輸量時，最高吞吐量在傳輸量為 0.2 mb 時，AODV 的最高吞吐量最少，GGVR 次之，而 GAR 的最高吞吐量最多，三者皆有顯著差距。最高吞吐量在傳輸量為 0.4 mb 時，AODV 的平均吞吐量最少，GGVR 次之，但兩者差距不大。而 GAR 的最高吞吐量最多，且有顯著差距。最高吞吐量在傳輸量為 0.6 mb 時，AODV 的最高吞吐量最少，GGVR 次之，而 GAR 的最高吞吐量最多，三者皆有顯著差距。最高吞吐量在傳輸量為 0.8 mb 時，AODV 的最高吞吐量最少，GGVR 次之，兩者有些微差距。而 GAR 的最高吞吐量最多，且有顯著差距。最高吞吐量在傳輸量為 1.0 mb 時，AODV 的最高吞吐量最少，GGVR 次之，而 GAR 的最高吞吐量最多，三者皆有顯著差距。GGVR 的最高吞吐量在傳輸量的變化下，表現狀況穩定良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

第七節 實驗結果整理

表 4- 5 佇列網路效能測量表現最佳方法

Queue	封包遺失率	平均封包延遲時間	最高封包延遲時間	平均抖動率	最高抖動率	平均吞吐量	最高吞吐量
5	AODV	AODV	AODV	GGVR	GGVR	AODV	GGVR
10	AODV	GGVR	GGVR	GGVR	GGVR	AODV	GGVR
15	GGVR	GGVR	GGVR	GGVR	GGVR	AODV	GGVR
25	GGVR	GGVR	GGVR	GGVR	GGVR	GGVR	GGVR
50	GGVR	GGVR	GGVR	GGVR	GGVR	GGVR	GGVR
75	GGVR	GGVR	GGVR	GGVR	GGVR	GGVR	GGVR

GGVR 的封包遺失率在佇列為 15、25、50 和 75 時，封包遺失率較低。GGVR 的平均封包延遲時間在佇列為 10、15、25、50 和 75 時，平均封包延遲時間較少。GGVR 的最高封包延遲時間在佇列為 10、15、25、50 和 75 時，最高封包延遲時間較少。GGVR 的平均抖動率在佇列的變化下，平均抖動率較低，皆表現良好。GGVR 的最高抖動率在佇列的變化下，最高抖動率較低，表現良好。GGVR 的平均吞吐量在佇列為 25、50 和 75 時，平均吞吐量較高。GGVR 的最高吞吐量在佇列的變化下，最高吞吐量皆較高。可能是有效迴避空隙的路由方式，在此時發揮了效果。

表 4- 6 封包大小網路效能測量表現最佳方法

Packet size	封包遺失率	平均封包延遲時間	最高封包延遲時間	平均抖動率	最高抖動率	平均吞吐量	最高吞吐量
128	AODV	GGVR	AODV	GGVR	GGVR	GGVR	GGVR
256	GGVR	GGVR	GGVR	GGVR	AODV	AODV	GGVR
512	AODV	GGVR	GGVR	GGVR	GGVR	AODV	GGVR
1024	GGVR	GGVR	GGVR	GGVR	GGVR	AODV	GGVR

GGVR 的封包遺失率在封包大小為 256 bytes 與 1024 bytes 時，封包遺失率較低。GGVR 的平均封包延遲時間在封包大小的變化下，平均封包延遲時間皆較少，表現良好。GGVR 的最高封包延遲時間在封包大小的變化下，在封包大小為 256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好。GGVR 的平均抖動率在封包大小的變化下，在封包大小為 128 bytes、256 bytes、512 bytes 和 1024 bytes 時表現狀況良好。GGVR 的最高抖動率在封包大小的變化下，在封包大小為 128 bytes、512 bytes 和 1024 bytes 時表現狀況良好。GGVR 的平均吞吐量在封包大小的變化下，在封包大小為 128 bytes 時表現狀況良好。GGVR 的最高吞吐量在封包大小的變化下，表現狀況良好。可能是有效迴避空隙的路由方式，在此時發揮了效果。

表 4- 7 傳輸量網路效能測量表現最佳方法

Rate	封包遺失率	平均封包延遲時間	最高封包延遲時間	平均抖動率	最高抖動率	平均吞吐量	最高吞吐量
0.2	AODV	GGVR	GGVR	GGVR	GGVR	AODV	GAR
0.4	GGVR	GGVR	GGVR	GGVR	GAR	AODV	GAR
0.6	GGVR	GGVR	GGVR	GGVR	GAR	GGVR	GAR
0.8	GGVR	GGVR	GGVR	GGVR	GAR	GGVR	GAR
1.0	GGVR	GGVR	GGVR	GGVR	GAR	GGVR	GAR

在控制變因為傳輸量時，封包遺失率表現狀況良好，封包遺失率較低，在傳輸量的變化下，皆能有穩定，且良好的表現。GGVR 的平均封包延遲時間在傳輸量的變化下，平均封包延遲時間皆較少，表現狀況顯著良好。GGVR 的最高封包延遲時間在傳輸量的變化下，最高封包延遲時間皆較少，表現狀況顯著良好。GGVR 的平均抖動率在傳輸量的變化下，平均抖動率較低，表現狀況顯著良好。GGVR 的最高抖動率在傳輸量的變化下，在傳輸量為 0.2 mb 時表現良好。GGVR 的平均吞吐量在傳輸量的變化下，在傳輸量為 0.6 mb、0.8 mb 和 1.0 mb 時表現良好。

第五章 結論

隨著科技的日新月異，有一天任何通訊產品都可能被視為一個節點，如智慧型手機、平板電腦與電子書的日漸普及，行動裝置的數量與日俱增。所以如何能提供完善且穩定的連線品質是當前重要的課題。由於在無線網際網路的環境中，通訊空隙的問題在無線網路上一直是個不確定的因素，隨時可能在通訊上產生問題，遇到通訊空隙的狀況。通訊空隙形成的原因為，當發送節點未能找到下一個在其附近有效節點到達地理目標節點的現象，稱為通訊空隙。雖然某些密集部署的無線節點可以減少在通信空隙網路中發生的可能性，但一些數據封包仍然可能遇到障礙空隙的誘導與無線網路不可靠的邊界節點等等。因此如何能有效的迴避空隙的問題產生，設計一個有效和高效方式的地理路由空隙處理技術，提昇資料傳輸的品質就顯得相當的重要。

本研究所提出的貪婪式空隙迴避法(GGVR)，在實驗的模擬結果中，分別與 GAR 與 AODV 的網路效能測量比較下。在控制變因為傳輸量時，GGVR 的封包遺失率表現狀況良好，封包遺失率較低。GGVR 的平均封包延遲時間在佇列、封包大小與傳輸量的變化下，平均封包延遲時間皆較少，表現狀況顯著良好。GGVR 的最高封包延遲時間在佇列、封包大小與傳輸量的變化下，最高封包延遲時間普遍皆較少，表現狀況顯著良好。GGVR 的平均抖動率在佇列與傳輸量變化下，抖動狀況並不嚴重，表現良好。GGVR 的最高抖動率在佇列的變化下，抖動狀況並不嚴重，表現良好。GGVR 的平均吞吐量在佇列的變化下，平均吞吐量較高，表現較佳。GGVR 的最高吞吐量在佇列的變化下，最高吞吐量較高。GGVR 在網路效能測量的比較下，都有不錯的表現。證明本研究所提出的 GGVR 在有效迴避空隙，並在解決通訊空隙問題上，能有不錯的結果。

本研究所提出的貪婪式空隙迴避法，具備發現空隙與定義空隙，在資料傳遞時遇到空隙問題的狀況前能事先迴避空隙。並以空隙範圍切線來引導路由，減少

傳遞轉發的節點數，縮短路由距離，能更省時且有效率的轉送封包。並確保資料傳遞的穩定性，減少資料的重送與遺失，減少傳遞重複的資料，降低網路流量以達到節省能源的目的，解決目前無線感測網路的地理空隙問題。

參考文獻

[1] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking*, Aug. 2000, pp. 243–254.

[2] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Holes in Sensor Networks,"*Mobile Networks and Applications*, Vol. 11, No. 2, pp. 187–200, Apr. 2006.

[3] M. Aissani, A. Mellouk, N. Badache, and M. Djebbar, "A New Approach of Announcement and Avoiding Routing Voids in Wireless Sensor Networks," in *Proceeding of the IEEE Global Telecommunications Conference*, Nov. 2008, pp. 1–5.

[4] W-J. Liu and K-T. Feng, "Greedy Routing with Anti-Void Traversal for Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, Vol. 8, No. 7, pp. 910-922, July 2009.

[5] M.Aissani, A.Mellouk, N.Badache, and M.Djebbar, "A Preventive Rerouting Scheme for Avoiding Voids in Wireless Sensor Networks," in *Proceeding of the IEEE Global Telecommunications Conference*, Dec. 2009, pp. 1–5.

[6] H. Wang, X. Zhang, and A. Khokhar, "Efficient Void Handling in Contention-Based Geographic Routing for Wireless Sensor Networks," in *Proceeding of the IEEE Global Telecommunications Conference*, Nov. 2007, pp. 663–667.

[7] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks,"*Proc. IEEE Int'l Conf. Distributed Computing Systems*, May.2003, pp. 46-55.

[8] I. Stojmenovic, M. Russell, and B. Vukojevic, "Depth First Search and Location Based Localized Routing and QoS Routing in Wireless Networks," *Proc. IEEE Int'l Conf. Parallel Processing*, 2000, pp. 173-180.

[9] W. Jia, T. Wang, G. Wang, and M. Guo, "Hole Avoiding in Advance Routing in Wireless Sensor Networks," *Proc. IEEE WCNC 2007*, March. 2007, pp. 3519 - 3523.

[10] M. Aissani, A. Mellouk, N. Badache, and B. Saidani, "Oriented Void Avoidance Scheme for Real-Time Routing Protocols in Wireless Sensor Networks," *IEEE Global Communications Conference*, Nov. 2008, pp.1-5.

[11] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," *IEEE Trans. Mobile Comput.*, Vol. 5, No. 6, June 2006, pp. 738–754.

[12] S. Chen, G. Fan, and J.-H. Cui, "Avoid 'void' in Geographic Routing for Data Aggregation in Sensor Networks," *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, Vol. 2, No. 1, July 2006, pp. 169-178.

[13] T. Roosta , M. Menzo, and S. Sastry, "Probabilistic Geographic Routing Protocol for Ad Hoc and Sensor Networks," *in International WorkShop on Wireless AdHoc Networks (IWWAN)* ,May 2005.

[14] Y. Xu, W. C. Lee, J. Xu, and G. Mitchell, "PSGR: Priority-based Stateless Geo-routing in Wireless Sensor Networks," *in IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Nov. 2005, 8 pp. - 680.

[15] J. Na, D. Soroker, and C. K. Kim, "Greedy geographic routing using dynamic potential field for wireless ad hoc networks," *IEEE Communications Letters*, Vol. 11, No. 3, March 2007, pp. 243-245.

[16] R. Tang, S. Guo , H. Ji, and C. Gong, "A Heuristic Optimization Algorithm For Geographic Greedy Hole-Bypassing Routing Algorithms in WMSNS," *Proceedings of Broadband Network and Multimedia Technology (IC-BNMT)*, Oct. 2010, pp. 540-545.

[17] P.Pranitha , G.Swamy , and A. Manjula, "A Review on Enhanced GPSR protocol For Wireless Sensor Networks," *Computer Engineering and Intelligent Systems*, Vol. 2, No.4, 2011.

[18] A. Baadache, and A. Belmehdi, "Avoiding Black hole and Cooperative Black hole Attacks in Wireless Ad hoc Networks," *International Journal of Computer Science and Information Security*, Vol. 7, No. 1, 2010.

[19] G-Y. Chang, J-P. Sheu, C-W. Chen, S-Y. Wang, and C-C. Chang, "Contour-Based Hole Avoiding Routing Protocol in Wireless Sensor Networks," *IEEE Communications Society subject matter experts for publication in the IEEE Globecom*, 2010.

[20] S. Xia, X. Yin, H. Wu, M. Jin, and X. Gu, "Deterministic Greedy Routing with Guaranteed Delivery in 3D Wireless Sensor Networks," *in Proc. of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2011.

[21] N. Le, N. Hieu, B. Quan, N. Van, "Efficient Approximation of Routing Holes in Wireless Sensor Networks," *Proceedings of the Second Symposium on Information and Communication Technology*, 2011.

[22] Z. Wang, D. Zhanga, O. Alfandi, and D. Hogrefe, “Efficient Geographical 3D Routing for Wireless Sensor Networks in Smart Spaces,” *Baltic Congress on Future Internet and Communications*, 2011.

[23] H. Choo, M. Choi, M. Shon, “Efficient Hole Bypass Routing Scheme Using Observer Packets for Geographic Routing in Wireless Sensor Networks,” *ACM SIGAPP Applied Computing Review*, Vol.11,No.4, 2011, pp. 7-16.

[24] S-F. Hwang, C-H. Yang, Y-Y. Su, C-R. Dow, “Energy Efficient Hole Bypassing Routing in Wireless Sensor Networks,” *Computer Science and Information Technology (ICCSIT)* , July 2010, pp. 576-580.

[25] J. Jia, J. Chen, X. Wang, and L. Zhao, “Energy-Balanced Density Control to Avoid Energy Hole for Wireless Sensor Networks,” *International Journal of Distributed Sensor Networks*, 2012.

[26] Y. Liu, L-X. Cai, X. Shen, and J-W. Mark, “Exploiting Heterogeneity Wireless Channels for Opportunistic Routing in Dynamic Spectrum Access Networks,” *IEEE Communications Society subject matter experts for publication in the IEEE ICC*, 2011, pp. 1-5.

[27] N-D. Nguyen, D-Tu. Nguyen, M-A- L. Gall, N. Saxena, and H. Choo, “Greedy Forwarding with Virtual Destination Strategy for Geographic Routing in Wireless Sensor Networks,” *International Conference of Computational Science and Its Applications*, 2010, pp. 217-221.

[28] B.S. Raja, N.Prabakaran, V.R.S. Dhulipala, “Modified GPSR Based Optimal Routing Algorithm for Reliable Communication in WSNs,” *International Conference on Devices and Communications (ICDeCom)*, 2011, pp. 1-5.

[29] L. Nithyanandan, G. Sivarajesh, and P. Dananjayan, “Modified GPSR Protocol for Wireless Sensor Networks,” *International Journal of Computer and Electrical Engineering*, Vol. 2, No. 2, April 2010, pp. 324-328.

[30] C-H. Lin, S-A. Yuan, S-W. Chiu, and M-J. Tsai, “ProgressFace: An Algorithm to Improve Routing Efficiency of GPSR-like Routing Protocols in Wireless Ad Hoc Networks,” *IEEE Transactions on Computers* , Vol. 59, No. 6, 2010, pp. 822-834.

[31] P. Samundiswary, D.Sathian1, and P. Dananjayan, “Secured Greedy Perimeter Stateless Routing for Wireless Sensor Networks,” *International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC)* , Vol. 1, No. 2, June 2010.

[32] Z. Wang, X. Qian, X. Zhao, “The Strategies of Avoiding Energy Holes in Wireless Sensor Networks,” 4th International Congress on Image and Signal Processing (CISP), Vol. 5, 2011, pp. 2687-2691.

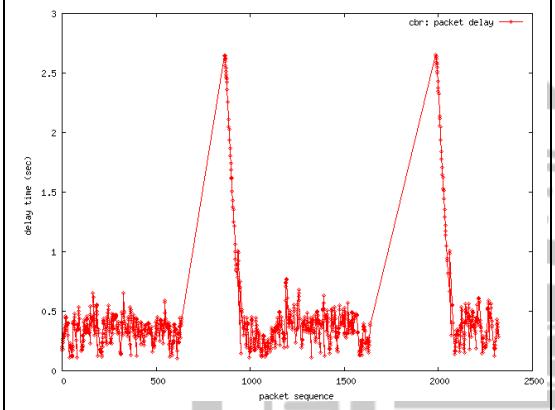
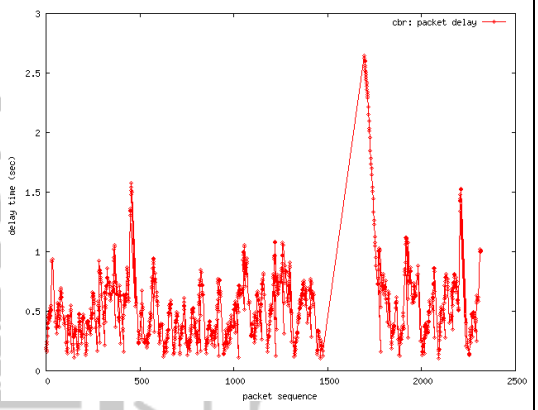
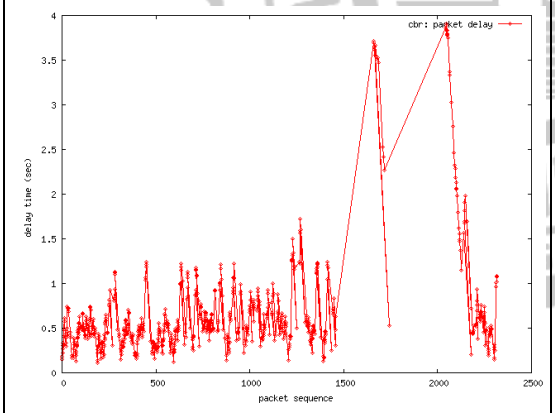
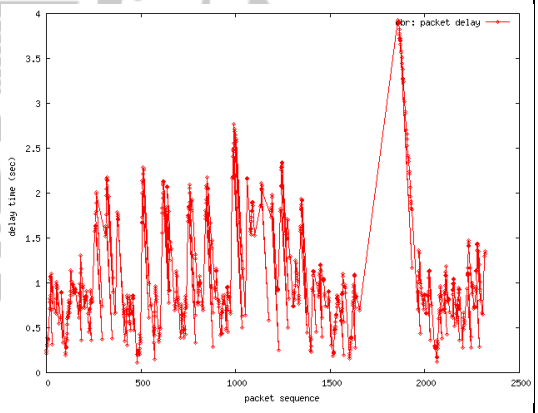
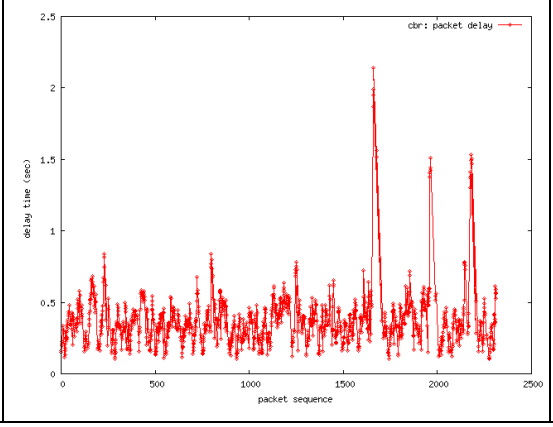
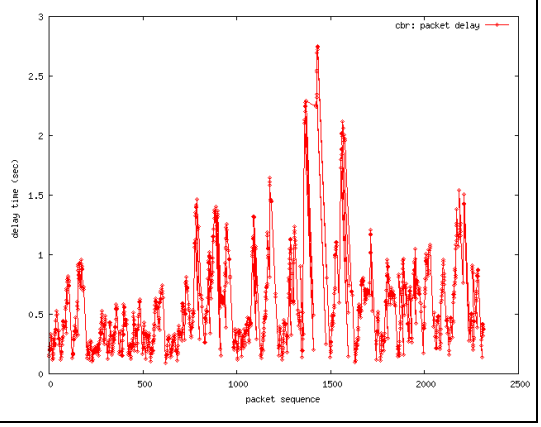
[33] J. You, Q. Han, D. Lieckfeldt, J. Salzmann, and D. Timmermann, “Virtual Position Based Geographic Routing for Wireless Sensor Networks,” *Computer Communications*, Vol. 33, July 2010, pp. 1255-1265.

附錄

NS2 實驗模擬的封包延遲詳細數據資料，封包延遲（packet dealy）即為封包到達時間與封包傳送時間的差值。

I. 佇列封包延遲（packet dealy）

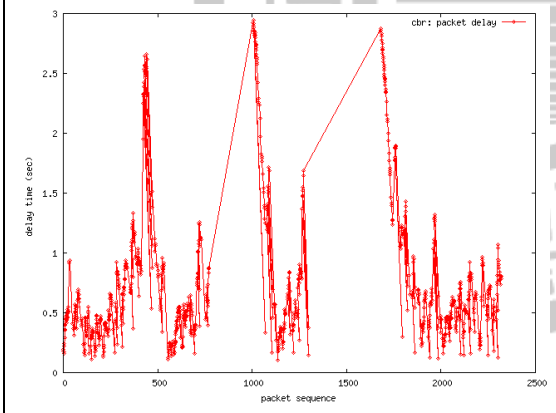
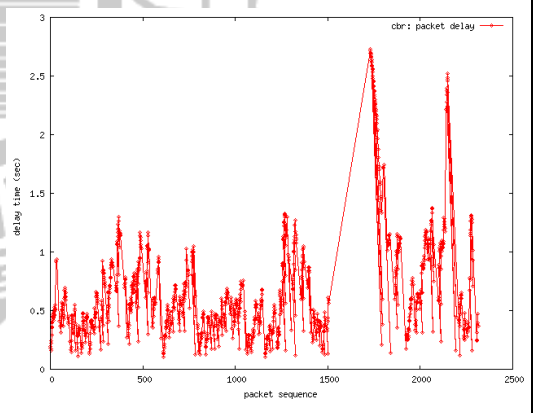
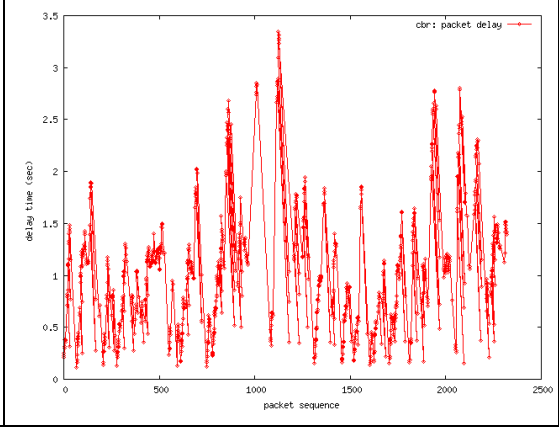
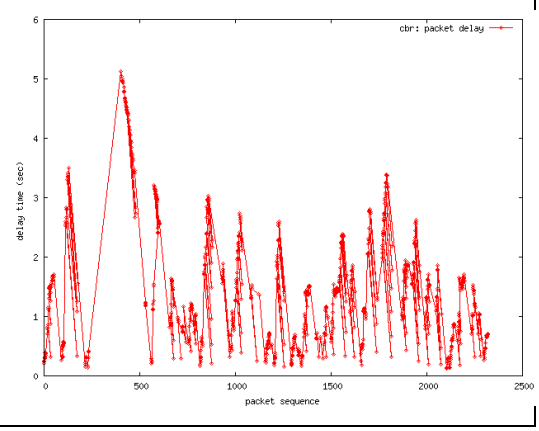
表 6 - 1 佇列為 5 與 10 的封包延遲時間

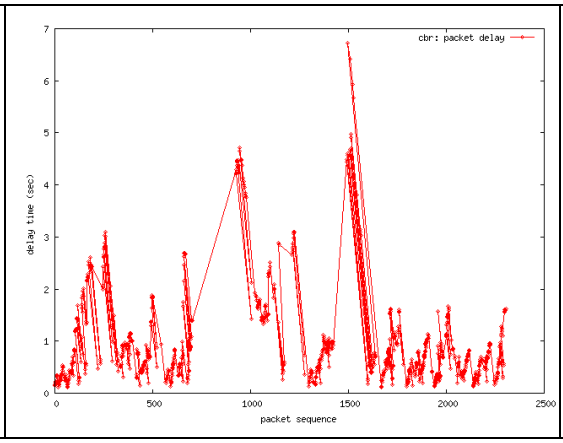
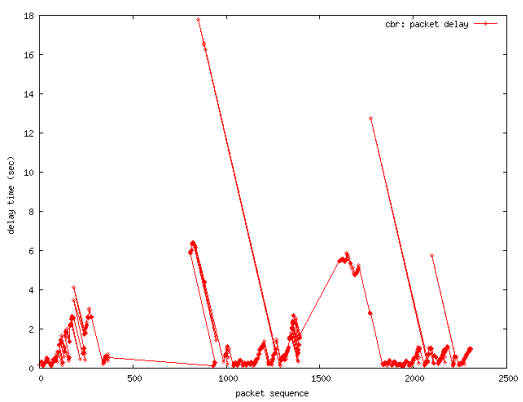
	佇列 5	佇列 10
GGVR		
GAR		
AODV		

平均封包延遲時間在佇列為 5 時，AODV 的平均封包延遲時間最少，GGVR 次之，而 GAR 的平均封包延遲時間最多。最高封包延遲時間在佇列為 5 時，AODV 的最高封包延遲時間最少，GGVR 次之，而 GAR 的最高封包延遲時間最多，但三者差距不大。在控制變因佇列為 5 時，GGVR 的封包延遲狀況並不嚴重，表現狀況良好。

平均封包延遲時間在佇列為 10 時，GGVR 的平均封包延遲時間最少，AODV 次之，但差距甚小。而 GAR 的平均封包延遲時間最多，且高出甚多。最高封包延遲時間在佇列為 10 時，GGVR 的最高封包延遲時間最少，AODV 次之，但差距甚小。而 GAR 的最高封包延遲時間最多，有些微差距。在控制變因佇列為 10 時，GGVR 的封包延遲狀況並不嚴重，表現狀況良好。

表 6 - 2 佇列為 15 與 25 的封包延遲時間

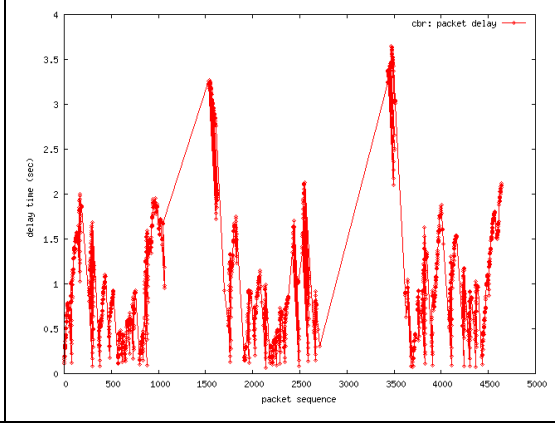
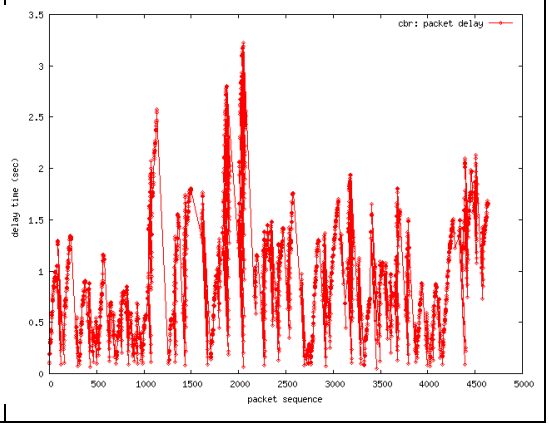
	佇列 15	佇列 25
GGVR		
GAR		

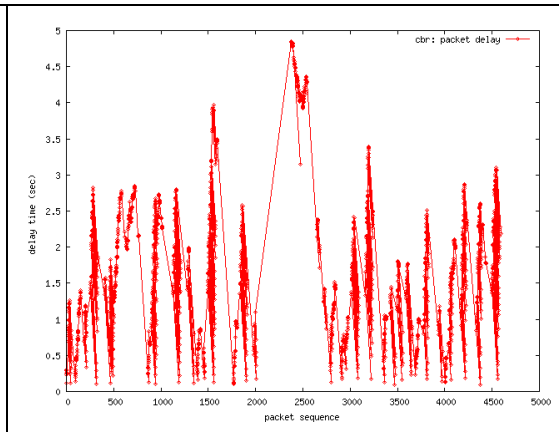
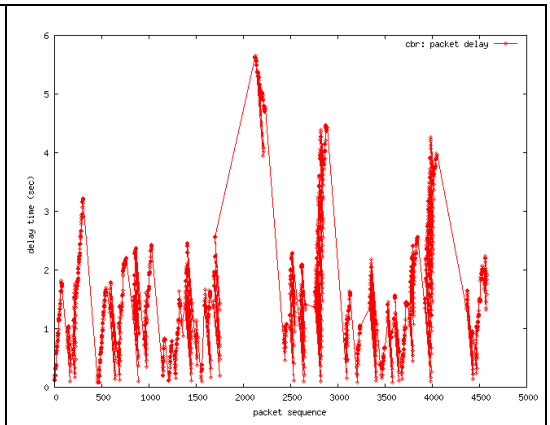
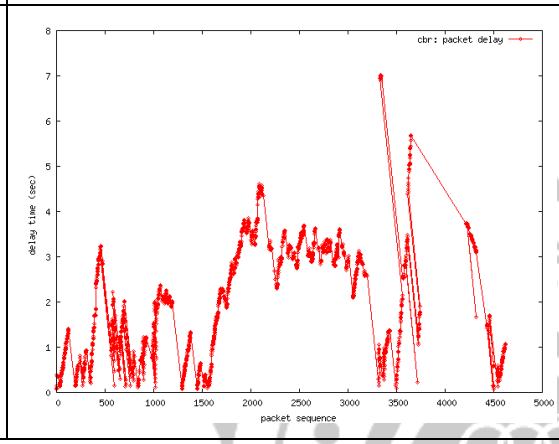
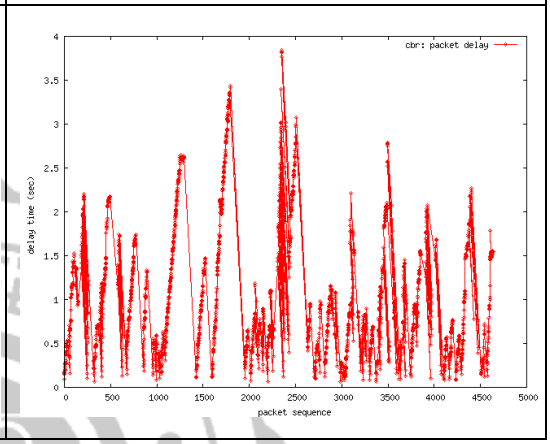
AODV		

平均封包延遲時間在佇列為 15 時，GGVR 的平均封包延遲時間最少，且有些為差距。而 AODV 次之，GAR 的平均封包延遲時間最多，但兩者差距甚小。最高封包延遲時間在佇列為 15 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者差距甚小。而 AODV 的最高封包延遲時間最多，且高出甚多。在控制變因佇列為 15 時，GGVR 的封包延遲狀況並不嚴重，表現狀況較佳。

平均封包延遲時間在佇列為 25 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的平均封包延遲時間最多，但兩者差距甚小。最高封包延遲時間在佇列為 25 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者有顯著差距。而 AODV 的最高封包延遲時間最多，差距甚多。在控制變因佇列為 25 時，GGVR 的封包延遲狀況並不嚴重，表現狀況顯著較佳。

表 6 - 3 佇列為 50 與 75 的封包延遲時間

	佇列 50	佇列 75
GGVR		

GAR		
AODV		

平均封包延遲時間在佇列為 50 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的平均封包延遲時間最多。最高封包延遲時間在佇列為 50 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者差距甚小。而 AODV 的最高封包延遲時間最多，有些微差距。在控制變因佇列為 50 時，GGVR 的封包延遲狀況並不嚴重，表現狀況越顯著良好。

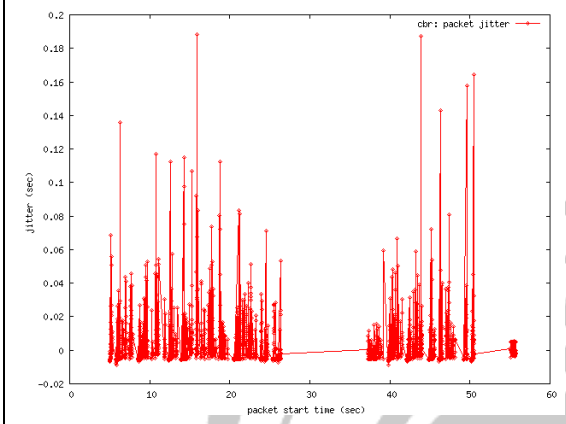
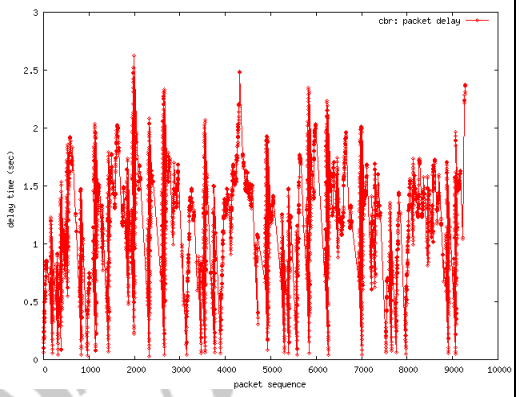
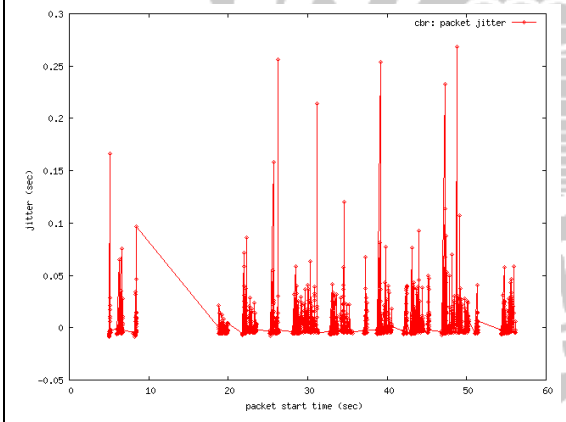
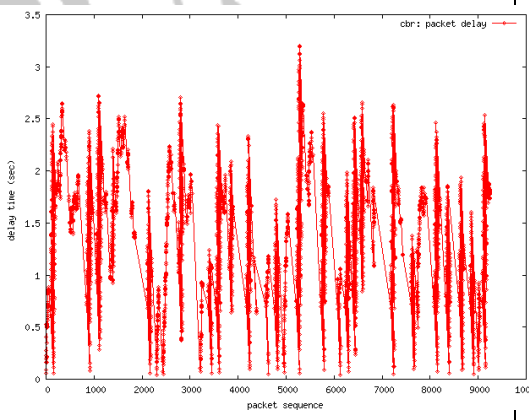
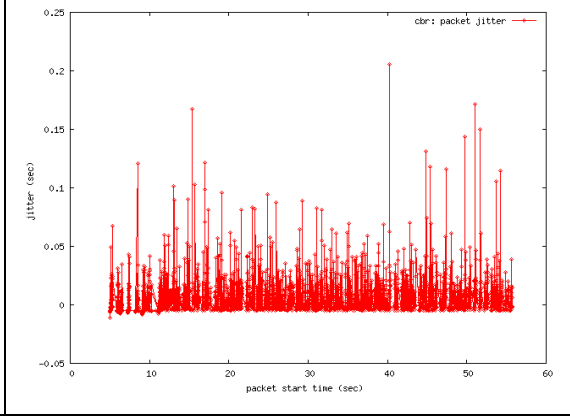
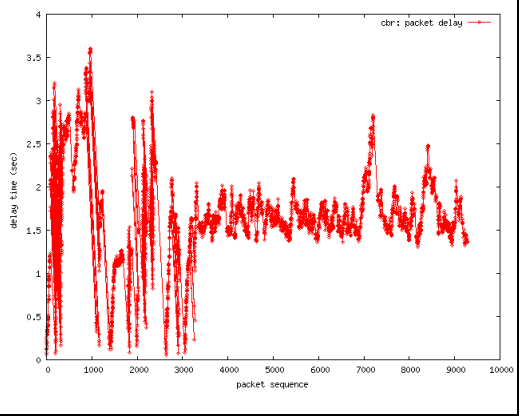
平均封包延遲時間在佇列為 75 時，GGVR 的平均封包延遲時間最少，AODV 次之，但兩者差距甚小。而 GAR 的平均封包延遲時間最多，且有顯著差距。最高封包延遲時間在佇列為 75 時，GGVR 的最高封包延遲時間最少，GAR 次之，兩者有顯著差距甚小。而 AODV 的最高封包延遲時間最多，且有顯著差距。在控制變因佇列為 75 時，GGVR 的封包延遲狀況並不嚴重，表現狀況越顯著良好。

GGVR 的平均封包延遲時間在佇列為 10、15、25、50 和 75 時，平均封包延遲時間較少。尤其在佇列為 15、25 與 50 時表現狀況顯著良好。GGVR 的最高封包延遲時間在佇列為 10、15、25、50 和 75 時，最高封包延遲時間較少。尤其在

佇列為 25、50 與 75 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

II. 封包大小封包延遲（packet delay）

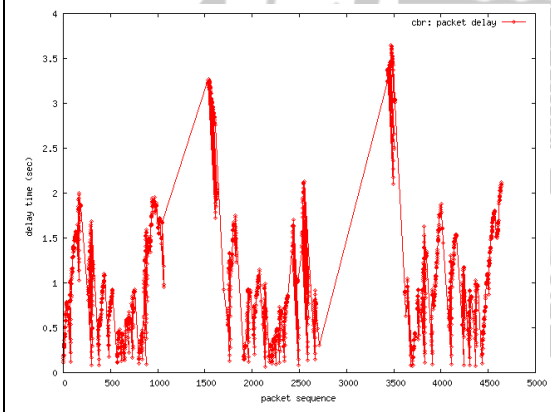
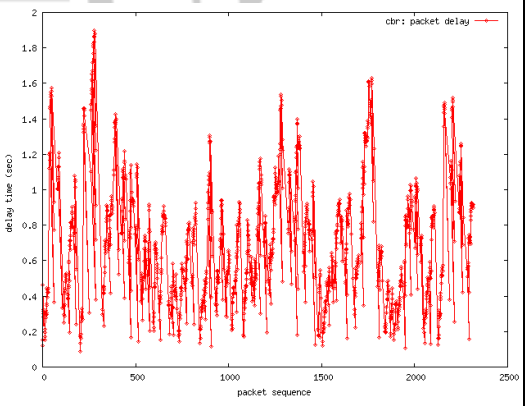
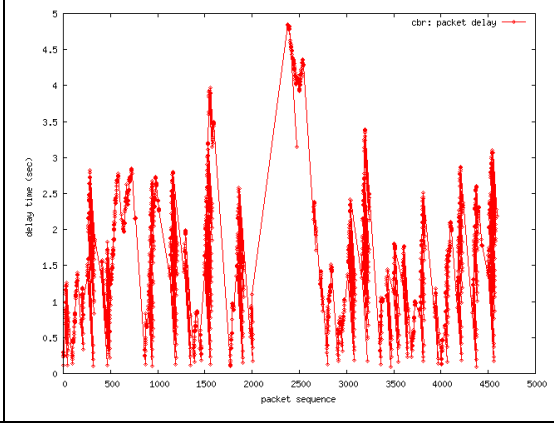
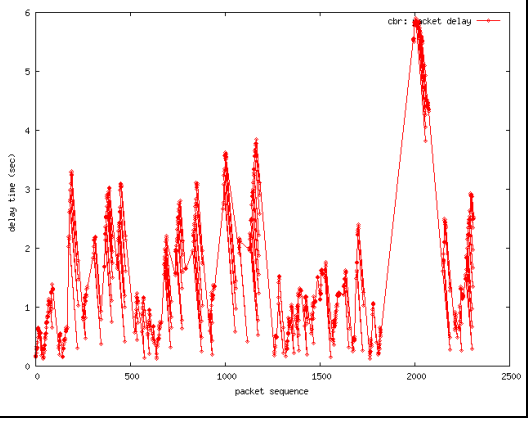
表 6 - 4 封包大小為 128 bytes 與 256 bytes 的封包延遲時間

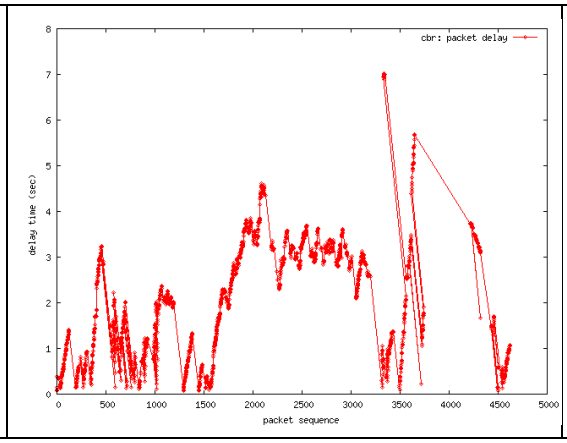
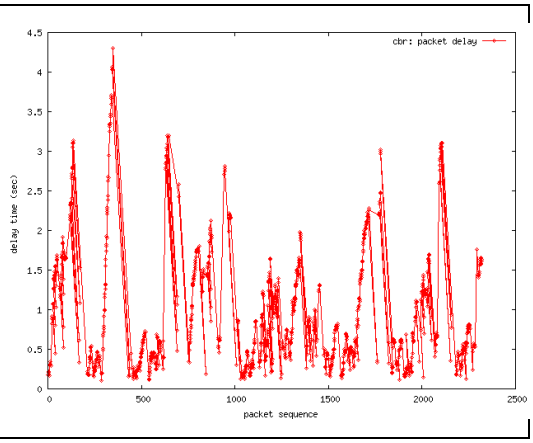
	封包大小 128 bytes	封包大小 256 bytes
GGVR		
GAR		
AODV		

平均封包延遲時間在封包大小為 128 bytes 時，GGVR、GAR 和 AODV 三者的平均封包延遲時間皆不高，皆有良好表現。平均封包延遲時間在封包大小為 128 bytes 時，GGVR、GAR 和 AODV 三者的平均封包延遲時間皆不高，皆有良好表現。最高封包延遲時間在封包大小為 128 bytes 時，GGVR、GAR 和 AODV 三者的最高封包延遲時間皆不高，皆有良好表現。在控制變因封包大小為 128 時，GGVR 的封包延遲狀況並不嚴重，表現狀況良好。

平均封包延遲時間在封包大小為 256 bytes 時，GGVR 的平均封包延遲時間最少，GAR 次之，而 AODV 的平均封包延遲時間最多，皆有些微差距。最高封包延遲時間在封包大小為 256 bytes 時，GGVR 的最高封包延遲時間最少，GAR 次之，而 AODV 的最高封包延遲時間最多，皆有些微差距。在控制變因封包大小為 256 時，GGVR 的封包延遲狀況並不嚴重，表現狀況顯著良好。

表 6 - 5 封包大小為 512 bytes 與 1024 bytes 的封包延遲時間

	封包大小 512 bytes	封包大小 1024 bytes
GGVR		
GAR		

AODV		

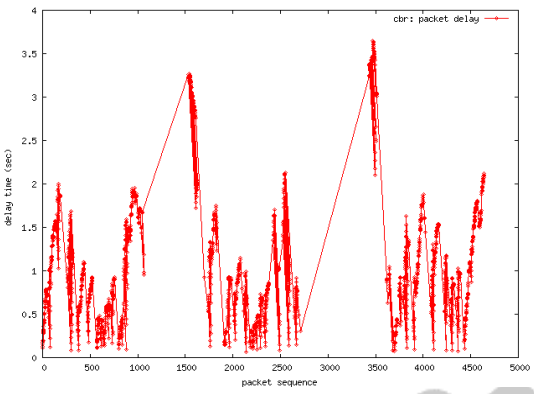
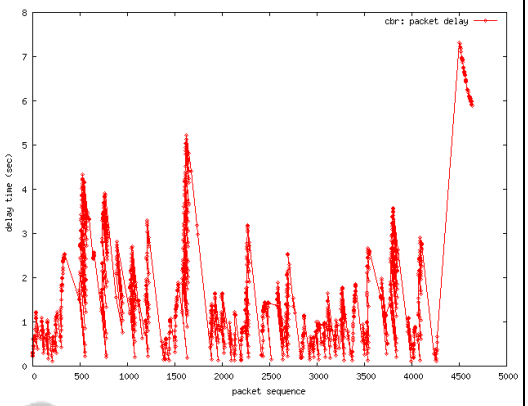
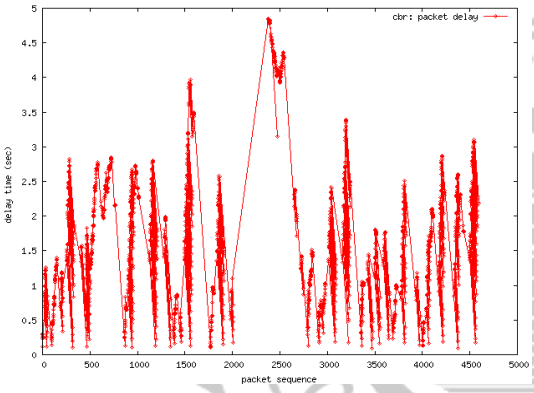
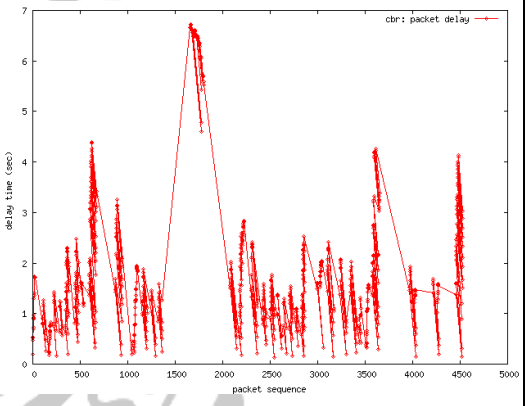
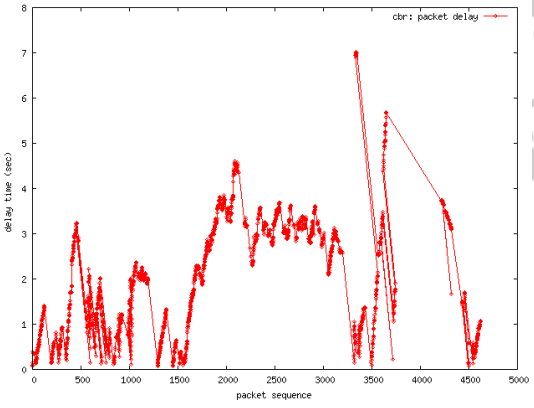
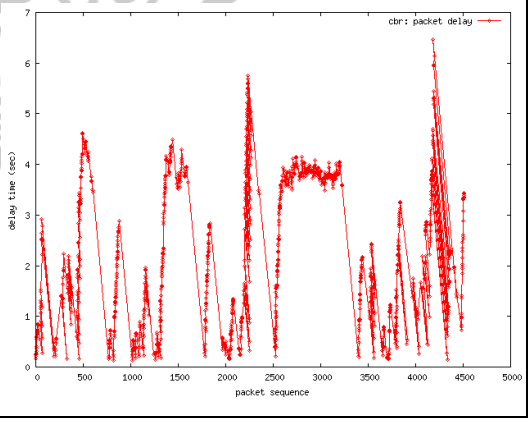
平均封包延遲時間在封包大小為 512 bytes 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，且有些微差距。最高封包延遲時間在封包大小為 512 bytes 時，GGVR 的最高封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的最高封包延遲時間最多，且差距甚大。在控制變因封包大小為 512 時，GGVR 的封包延遲狀況並不嚴重，表現狀況顯著良好。

平均封包延遲時間在封包大小為 1024 bytes 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的平均封包延遲時間最多，且有顯著差距。最高封包延遲時間在封包大小為 1024 bytes 時，GGVR 的最高封包延遲時間最少，且有顯著差距。而 AODV 次之，GAR 的最高封包延遲時間最多，且有顯著差距。在控制變因封包大小為 1024 時，GGVR 的封包延遲狀況並不嚴重，表現狀況顯著良好。

GGVR 的平均封包延遲時間在封包大小的變化下，平均封包延遲時間皆較少，表現良好。尤其在封包大小為 256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好。GGVR 的最高封包延遲時間在封包大小的變化下，在封包大小為 256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

III. 傳輸量封包延遲（packet delay）

表 6 - 6 傳輸量為 0.2 mb 與 0.4 mb 的封包延遲時間

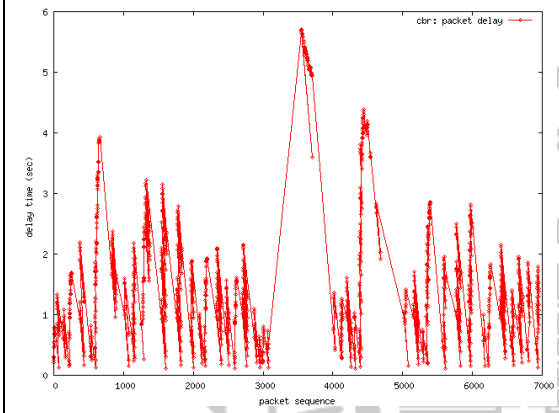
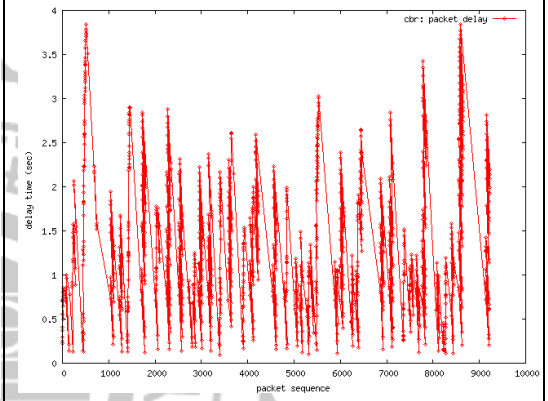
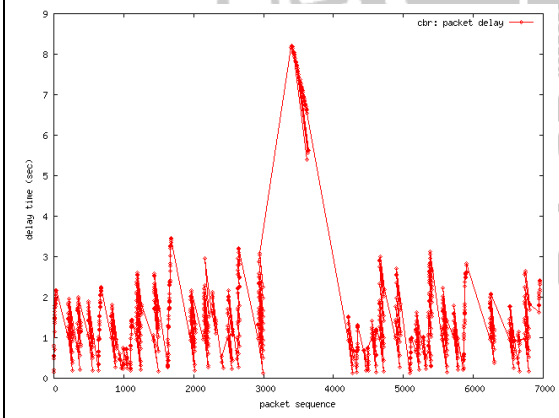
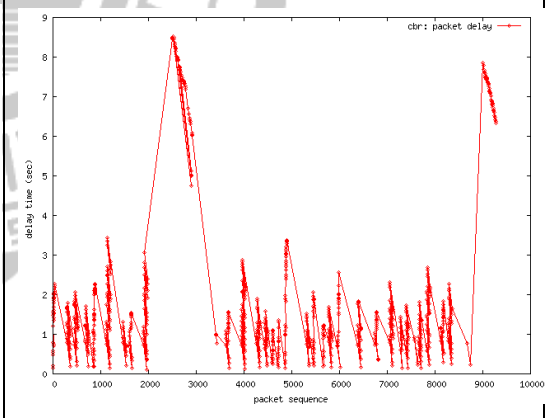
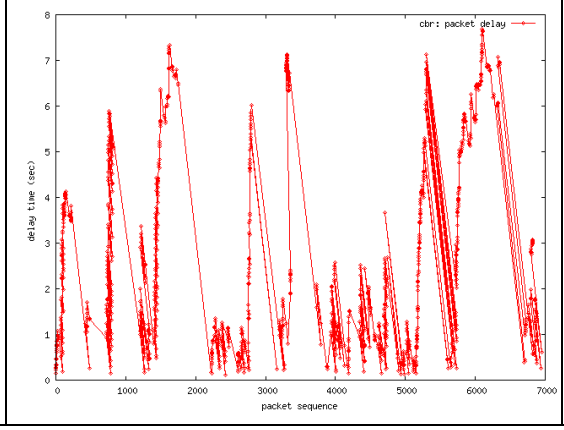
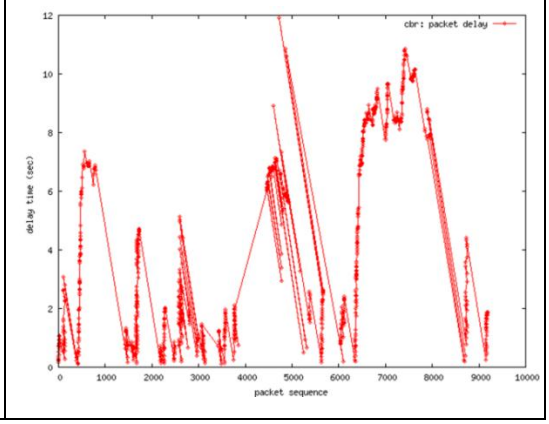
	傳輸量 0.2 mb	傳輸量 0.4 mb
GGVR		
GAR		
AODV		

平均封包延遲時間在傳輸量為 0.2 mb 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，兩者差距甚小。最高封包延遲時間在傳輸量為 0.2 mb 時，GGVR 的最高封包延遲時間最少，GAR 次之，而 AODV 的最高封包延遲時間最多，三者皆有顯著差距。在控制變因傳輸

量為 0.2 時，GGVR 的封包延遲狀況並不嚴重，表現狀況越顯著良好。

平均封包延遲時間在傳輸量為 0.4 mb 時，GGVR 的平均封包延遲時間最少，GAR 次之，而 AODV 的平均封包延遲時間最多，皆有些微差距。最高封包延遲時間在傳輸量為 0.4 mb 時，GGVR 的最高封包延遲時間最少，AODV 次之，而 GAR 的最高封包延遲時間最多，但三者差距不大。在控制變因傳輸量為 0.4 時，GGVR 的封包延遲狀況並不嚴重，表現無顯著差異。

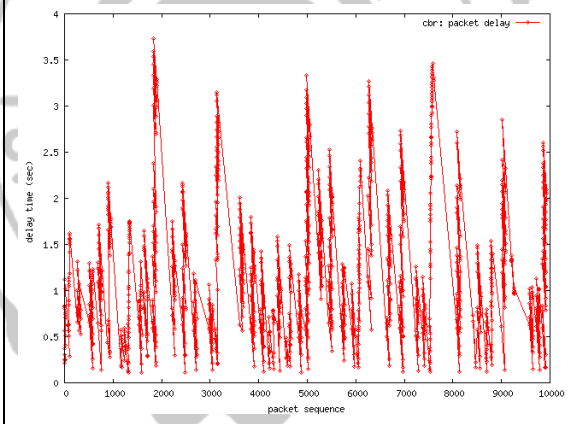
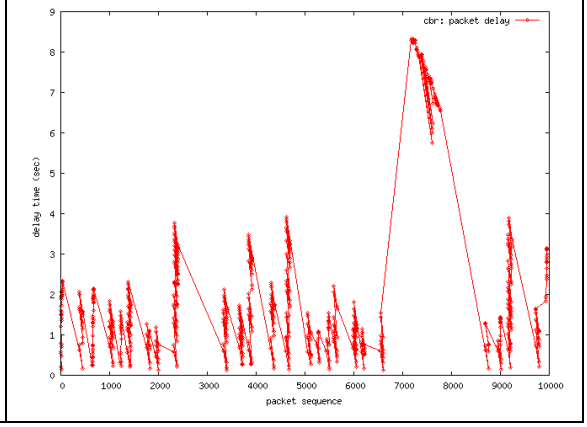
表 6 - 7 傳輸量為 0.6 mb 與 0.8 mb 的封包延遲時間

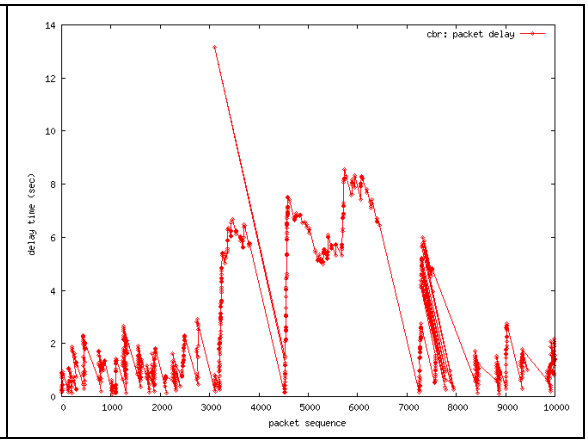
	傳輸量 0.6 mb	傳輸量 0.8 mb
GGVR		
GAR		
AODV		

平均封包延遲時間在傳輸量為 0.6 mb 時，GGVR 的平均封包延遲時間最少，GAR 次之，兩者有些微差距。而 AODV 的平均封包延遲時間最多，且有顯著差距。最高封包延遲時間在傳輸量為 0.6 mb 時，GGVR 的最高封包延遲時間最少，有顯著差距。而 AODV 次之，GAR 的最高封包延遲時間最多，兩者有些微差距。在控制變因傳輸量為 0.6 時，GGVR 的封包延遲狀況並不嚴重，表現較佳。

平均封包延遲時間在傳輸量為 0.8 mb 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，且有顯著差距。最高封包延遲時間在傳輸量為 0.8 mb 時，GGVR 的最高封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的最高封包延遲時間最多，且有顯著差距。在控制變因傳輸量為 0.8 時，GGVR 的封包延遲狀況並不嚴重，表現顯著良好。

表 6 - 8 傳輸量為 1.0 mb 的封包延遲時間

	傳輸量 1.0 mb
GGVR	
GAR	

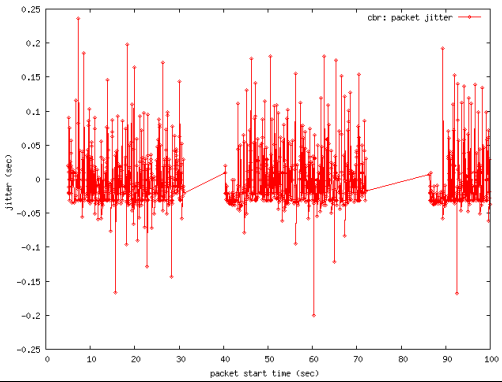
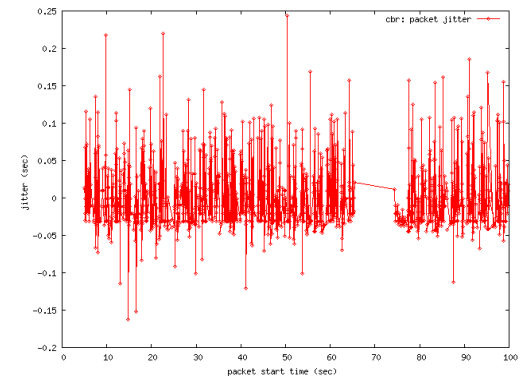
AODV	

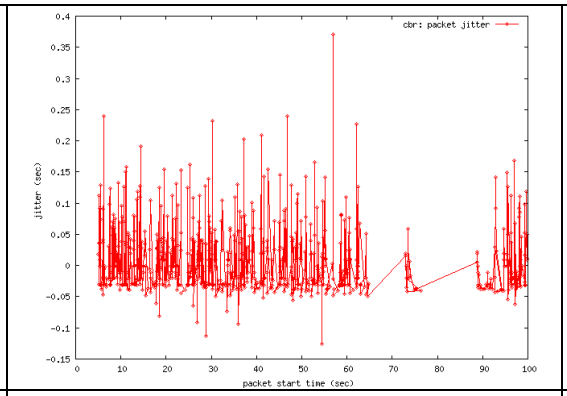
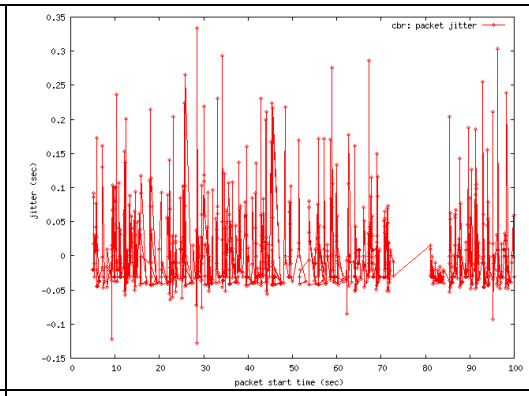
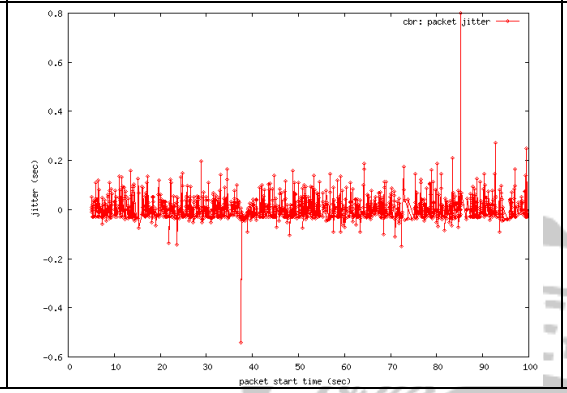
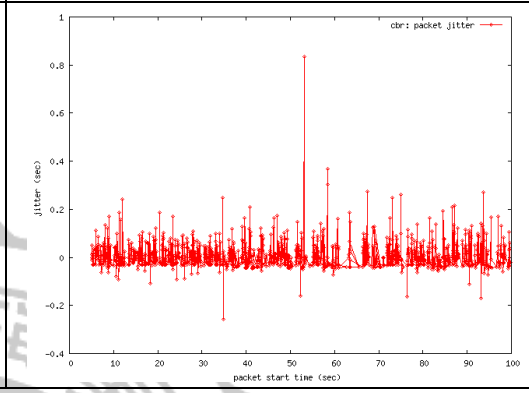
平均封包延遲時間在傳輸量為 1.0 mb 時，GGVR 的平均封包延遲時間最少，且有顯著差距。而 GAR 次之，AODV 的平均封包延遲時間最多，兩者有些微差距。最高封包延遲時間在傳輸量為 1.0 最高時，GGVR 的最高封包延遲時間最少，GAR 次之，而 AODV 的最高封包延遲時間最多，三者皆有顯著差距。在控制變因傳輸量為 1.0 時，GGVR 的封包延遲狀況並不嚴重，表現顯著良好。

GGVR 的平均封包延遲時間在傳輸量的變化下，平均封包延遲時間皆較少，表現良好。尤其在傳輸量為 0.2 mb、0.8 mb 和 1.0 mb 時，表現狀況顯著良好。GGVR 的最高封包延遲時間在傳輸量的變化下，在傳輸量為 0.2 mb、0.4 mb、0.6 mb、0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

IV. 佇列抖動率（packet jitter）

表 6 - 9 佇列為 5 與 10 的抖動率

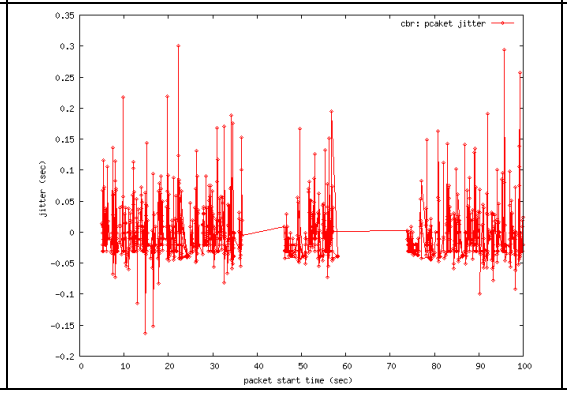
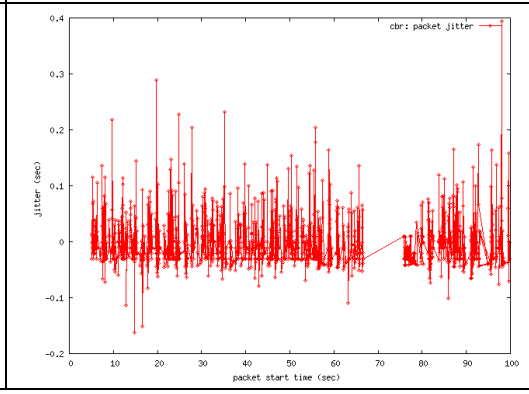
	佇列 5	佇列 10
GGVR		

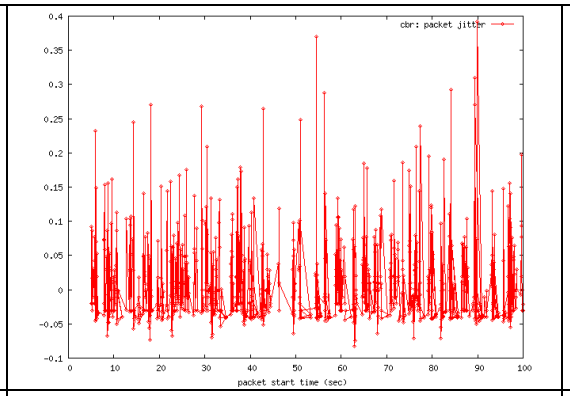
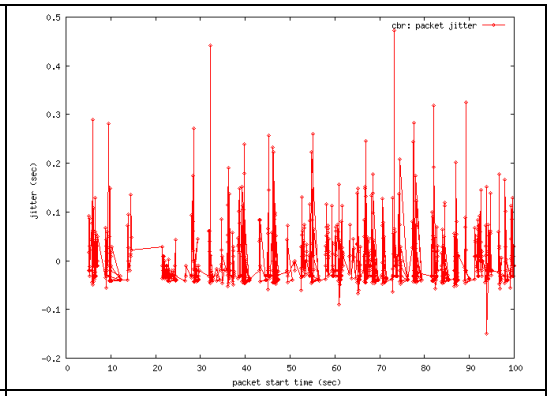
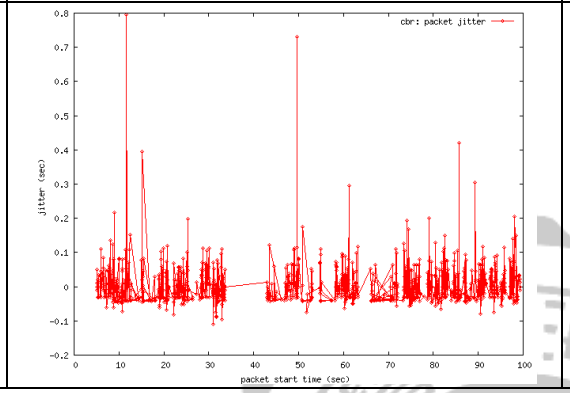
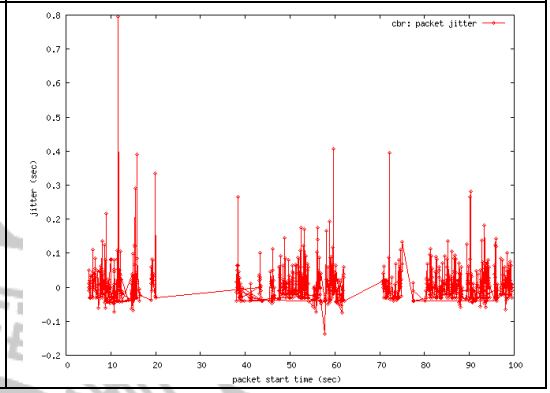
GAR		
AODV		

平均抖動率在佇列為 5 時，GGVR 的平均抖動率最少，且有顯著差距。而 AODV 次之，GAR 的平均抖動率最多，兩者有些微差距。最高抖動率在佇列為 5 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，且有顯著差距。

平均抖動率在佇列為 10 時，GGVR 的平均抖動率時間最少，且有顯著差距。而 GAR 次之，AODV 的平均抖動率最多，兩者有些微差距。最高抖動率在佇列為 10 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，有顯著差距。

表 6 - 10 佇列為 15 與 25 的抖動率

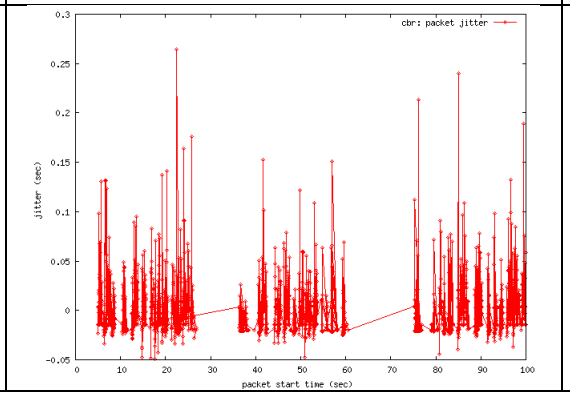
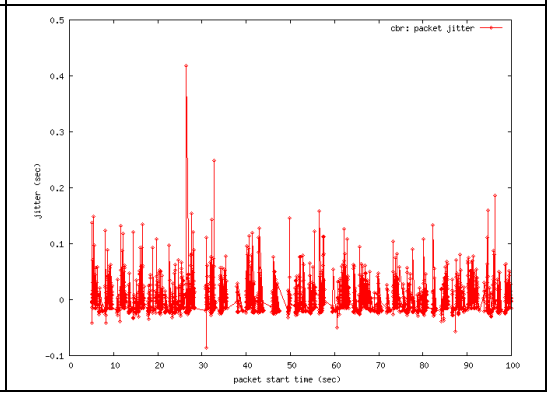
	佇列 15	佇列 25
GGVR		

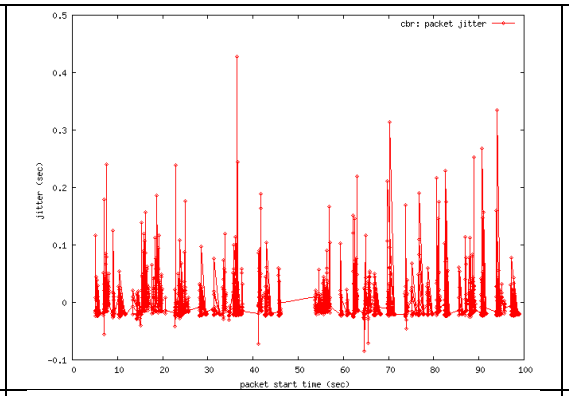
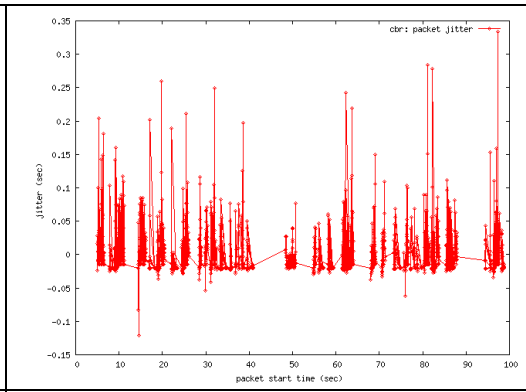
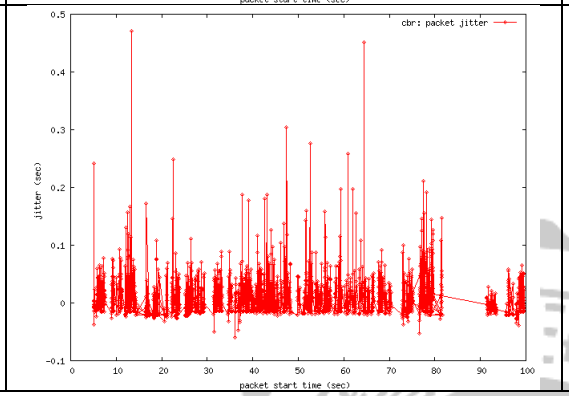
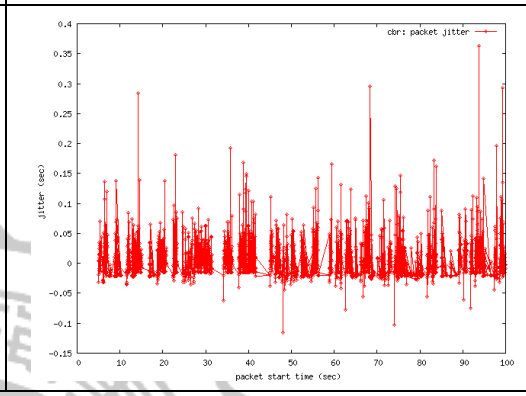
GAR		
AODV		

平均抖動率在佇列為 15 時，GGVR 的平均抖動率最少，且有顯著差距。而 AODV 次之，GAR 的平均抖動率最多，兩者有些微差距。最高抖動率在佇列為 15 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，且有顯著差距。

平均抖動率在佇列為 25 時，GGVR 的平均抖動率最少，且有顯著差距。而 GAR 次之，AODV 的平均抖動率最多，兩者有些微差距。最高抖動率在佇列為 25 時，GGVR 的最高抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的最高抖動率最多，且有顯著差距。

表 6 - 11 佇列為 50 與 75 的抖動率

	佇列 50	佇列 75
GGVR		

GAR		
AODV		

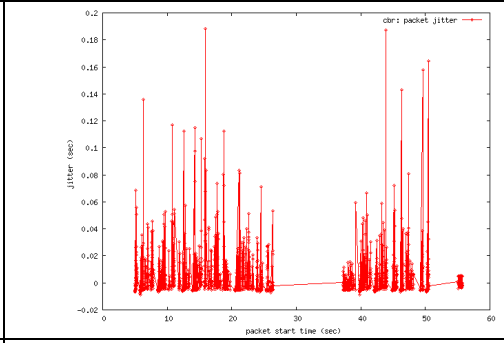
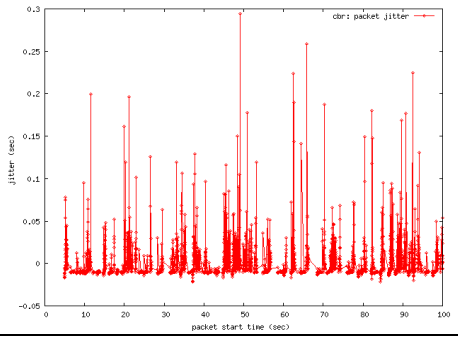
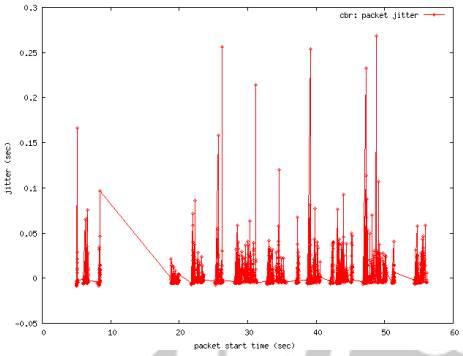
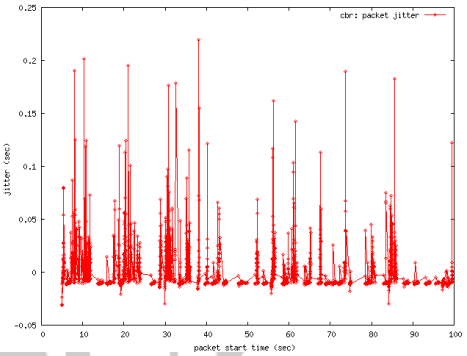
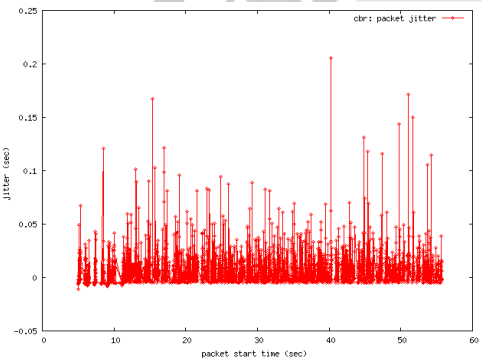
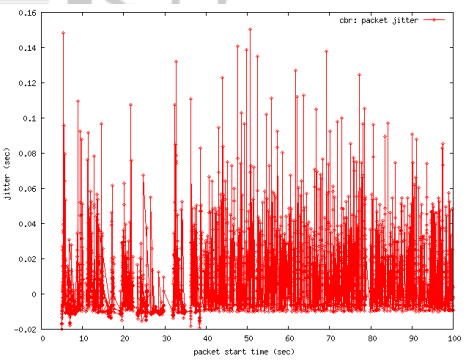
平均抖動率在佇列為 50 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，有顯著差距。最高抖動率在佇列為 50 時，GGVR 的最高抖動率最少，且有顯著差距。而 GAR 次之，AODV 的最高抖動率最多，兩者差距不大。

平均抖動率在佇列為 75 時，GGVR 的平均抖動率最少，GAR 次之，兩者有顯著差距。而 AODV 的平均抖動率最多，有些微差距。最高抖動率在佇列為 75 時，GAR 的最高抖動率最少，AODV 次之，而 GGVR 的最高抖動率最多，但三者差距不大。

GGVR 的抖動率在佇列的變化下，平均抖動率較少，表現狀況皆顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。GGVR 的最高抖動率在佇列為 5、10、15、25 和 50 時，最高抖動率較少。尤其在佇列為 50 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

V. 封包大小抖動率（packet jitter）

表 6 - 12 封包大小為 128 bytes 與 256 bytes 的抖動率

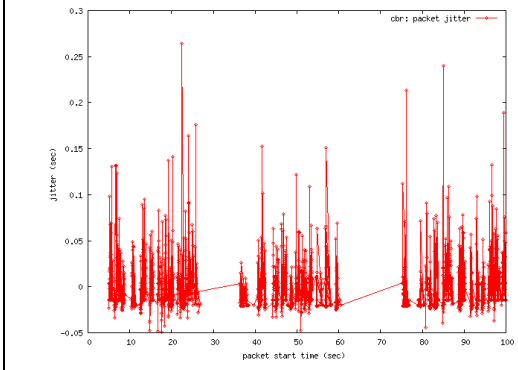
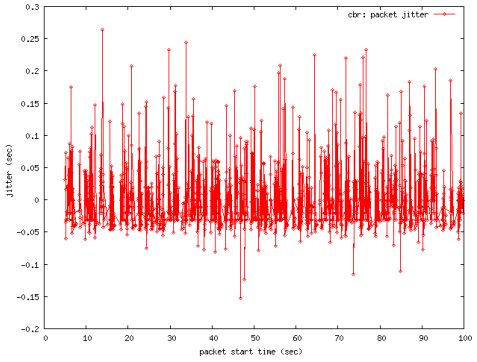
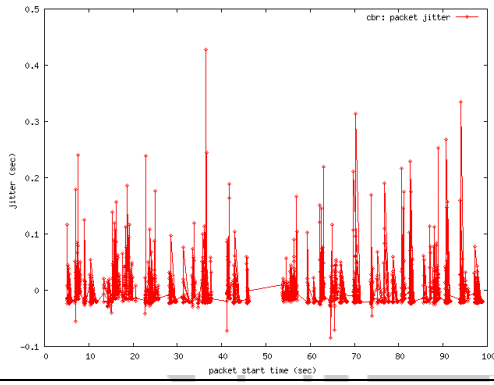
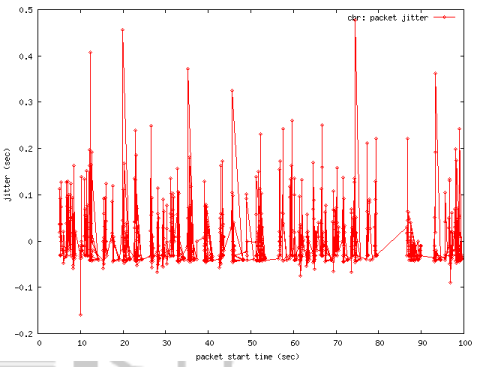
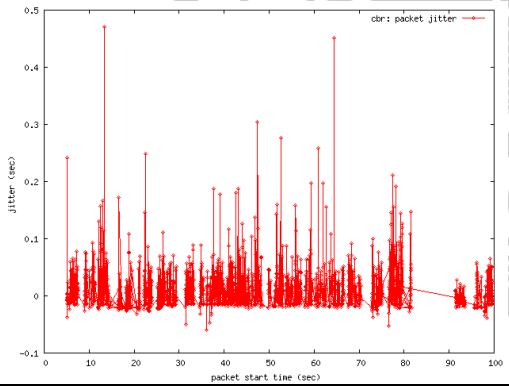
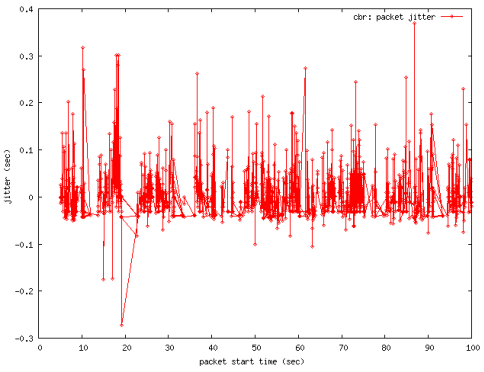
	封包大小 128 bytes	封包大小 256 bytes
GGVR		
GAR		
AODV		

平均抖動率在封包大小為 128 bytes 時， GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。最高抖動率在封包大小為 128 bytes 時，GGVR 的最高抖動率最少，AODV 次之，但兩者差距不大。而 GAR 的最高抖動率最多，有些微差距。

平均抖動率在封包大小為 256 bytes 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。最高抖動率在封包大小為 256 bytes 時，AODV 的最高抖動率最少，GAR 次之，而 GGVR 的

最高抖動率最多，三者皆有些微差距。

表 6 - 13 封包大小為 512 bytes 與 1024 bytes 的抖動率

	封包大小 512 bytes	封包大小 1024 bytes
GGVR		
GAR		
AODV		

平均抖動率在封包大小為 512 bytes 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。最高抖動率在封包大小為 512 bytes 時，GGVR 的最高抖動率最少，且有顯著差距。而 GAR 次之，AODV 的最高抖動率最多，兩者有些微差距。

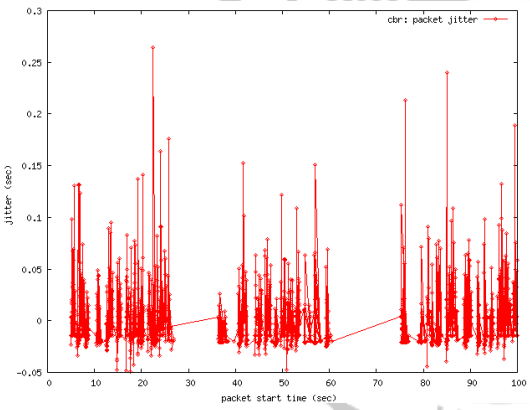
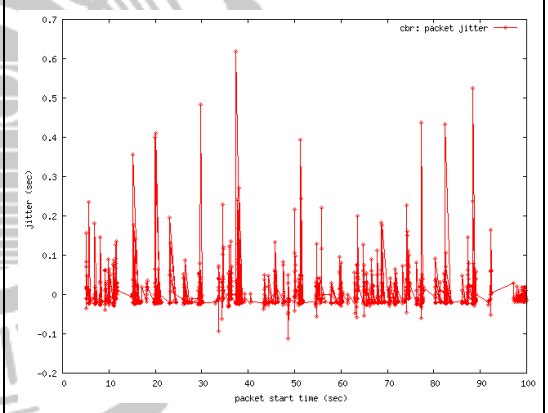
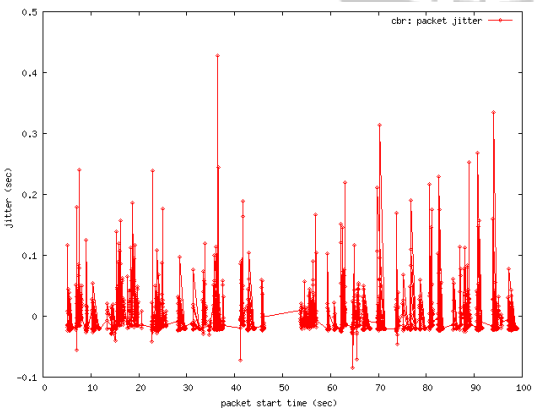
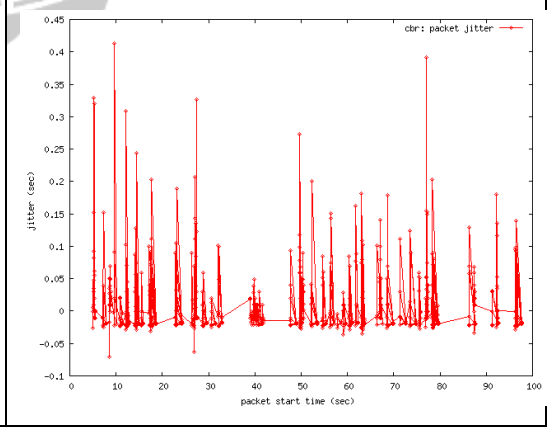
平均抖動率在封包大小為 1024 bytes 時，GGVR 的平均抖動率最少，GAR 次之，兩者有顯著差距。而 AODV 的平均抖動率最多，有些微差距。最高抖動率在

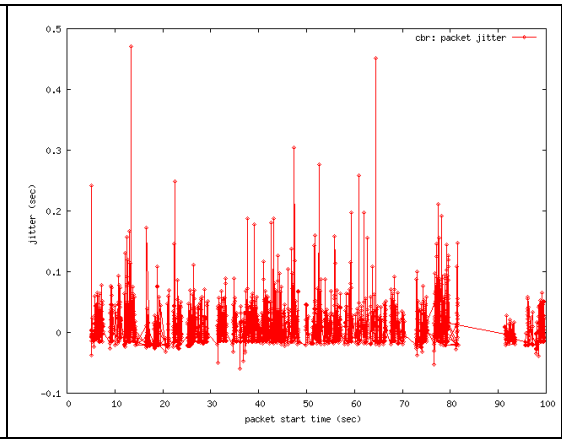
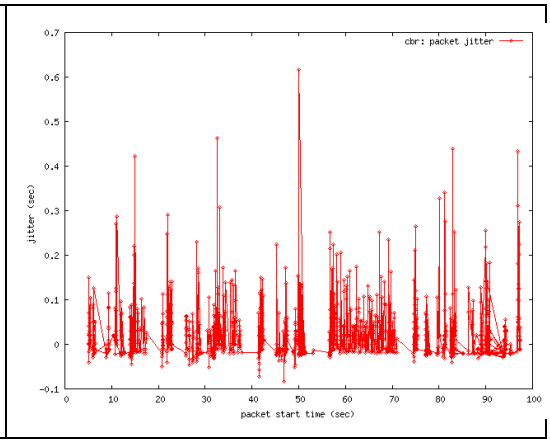
封包大小為 1024 bytes 時，GGVR 的最高抖動率最少，GAR 次之，而 AODV 的最高抖動率最多，三者皆有顯著差距。

GGVR 的平均抖動率在封包大小的變化下，平均抖動率皆較少，表現良好。尤其在封包大小為 128 bytes、256 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。GGVR 的最高封包延遲時間在封包大小的變化下，在封包大小為 128 bytes、512 bytes 與 1024 bytes 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

VI. 傳輸量抖動率（packet jitter）

表 6 - 14 傳輸量為 0.2 mb 與 0.4 mb 的抖動率

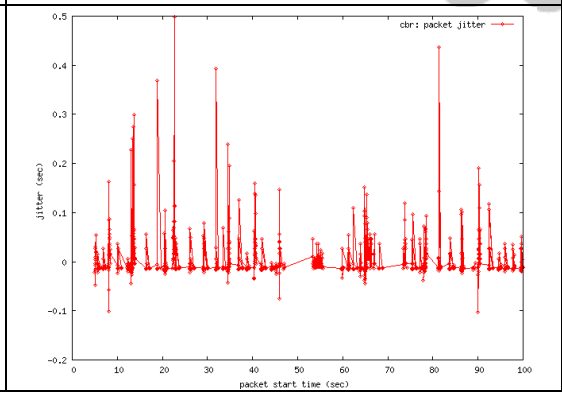
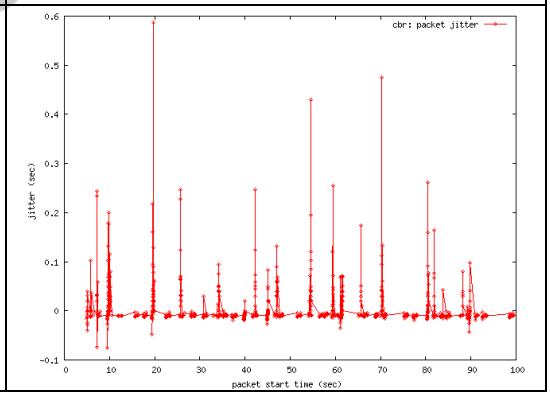
	傳輸量 0.2 mb	傳輸量 0.4 mb
GGVR		
GAR		

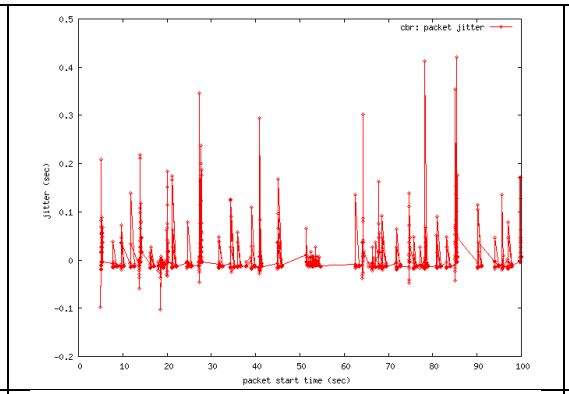
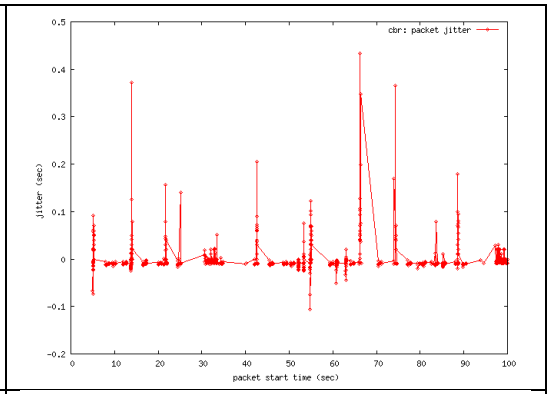
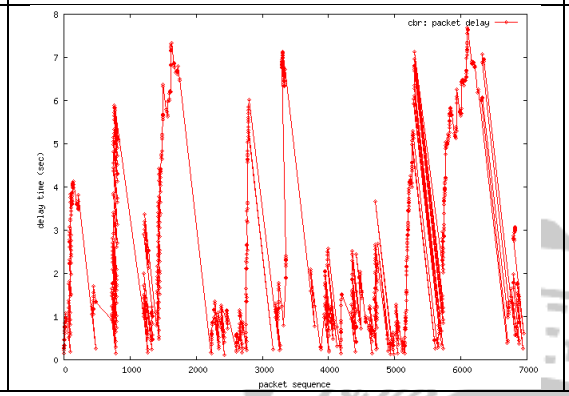
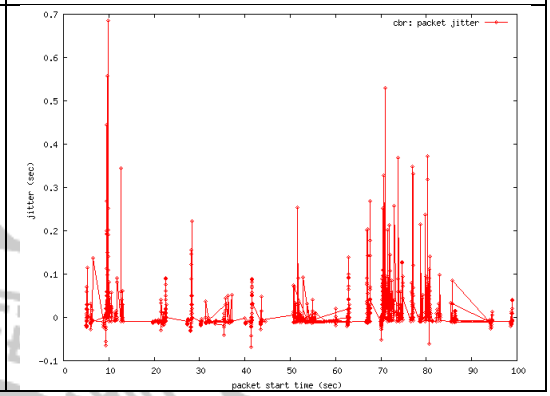
AODV		

平均抖動率在傳輸量為 0.2 mb 時，GGVR 的平均抖動率最少。而 GAR 次之，AODV 的平均抖動率最多，兩者差距顯著。最高抖動率在傳輸量為 0.2 mb 時，GGVR 的最高抖動率最少，有些微差距。而 GAR 次之，AODV 的最高抖動率最多，兩者只有些微差距。

平均抖動率在傳輸量為 0.4 mb 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。最高抖動率在傳輸量為 0.4 mb 時，GAR 的最高抖動率最少，且有顯著差距。而 AODV 次之，GGVR 的最高抖動率最多，兩者有些微差距。

表 6 - 15 傳輸量為 0.6 mb 與 0.8 mb 的抖動率

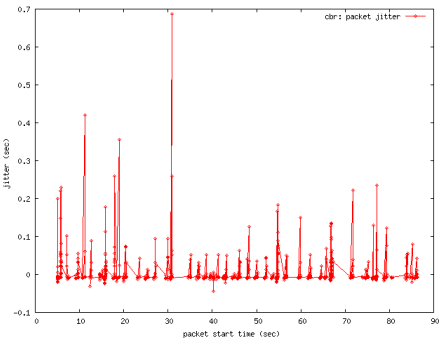
	傳輸量 0.6 mb	傳輸量 0.8 mb
GGVR		

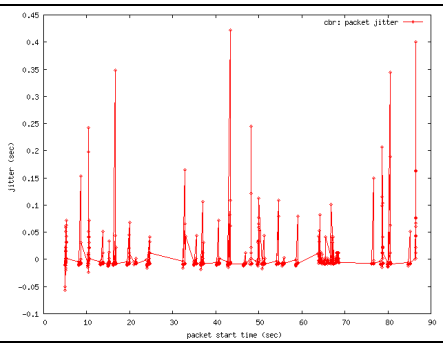
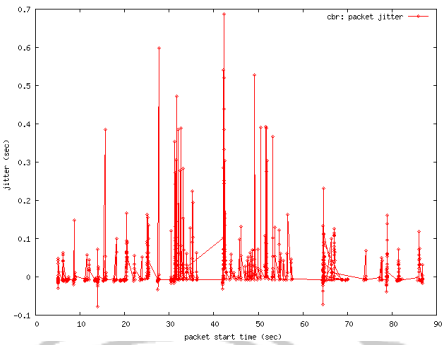
GAR		
AODV		

平均抖動率在傳輸量為 0.6 mb 時，GGVR 的平均抖動率最少，GAR 次之，而 AODV 的平均抖動率最多，三者皆有顯著差距。最高抖動率在傳輸量為 0.6 mb 時，GAR 的最高抖動率最少，GGVR 次之，而 AODV 的最高封包延遲時間最多，三者有顯著差距。

平均抖動率在傳輸量為 0.8 mb 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。最高抖動率在傳輸量為 0.8 mb 時，GAR 的最高抖動率最少，且有顯著差距。而 GGVR 次之，AODV 的最高抖動率最多，且有顯著差距。

表 6 - 16 傳輸量為 1.0 mb 的抖動率

	傳輸量 1.0 mb
GGVR	

GAR	
AODV	

平均抖動率在傳輸量為 1.0 mb 時，GGVR 的平均抖動率最少，GAR 次之，兩者有些微差距。而 AODV 的平均抖動率最多，且有顯著差距。最高抖動率在傳輸量為 1.0 最高時，GAR 的最高封包延遲時間最少，且有顯著差距。而 AODV 次之，GGVR 的最高抖動率最多，兩者差距不大。

GGVR 的平均抖動率在傳輸量的變化下，平均抖動率皆較少，表現良好。尤其在傳輸量為 0.4 mb、0.6 mb、0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。GGVR 的最高封包延遲時間在傳輸量的變化下，在傳輸量為 0.2 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

VII. 佇列吞吐量（throughput）

表 6 - 17 佇列吞吐量

Queue	5	10	15	25	50	75
GGVR	Average rate: 95197.2 Peak rate: 225504	Average rate: 101146.4 Peak rate: 250560	Average rate: 98662.7 Peak rate: 258912	Average rate: 110812.5 Peak rate: 275616	Average rate: 102621.2 Peak rate: 268128	Average rate: 110294.0 Peak rate: 257472
GAR	Average rate: 66810.3 Peak rate: 211984	Average rate: 75253.8 Peak rate: 233856	Average rate: 86271.4 Peak rate: 210336	Average rate: 80290.8 Peak rate: 215728	Average rate: 73213.7 Peak rate: 204288	Average rate: 75730.6 Peak rate: 204288
AODV	Average rate: 120007.7 Peak rate: 167040	Average rate: 113288.4 Peak rate: 167040	Average rate: 110897.1 Peak rate: 183744	Average rate: 99420.6 Peak rate: 183744	Average rate: 95915.8 Peak rate: 144704	Average rate: 99209.6 Peak rate: 148960

GGVR 的平均吞吐量在佇列為 25、50 和 75 時，平均吞吐量較高。尤其在佇列為 25 時表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。GGVR 的最高吞吐量在佇列為 5、10、15、25、50 和 75 時，最高吞吐量較高。尤其在佇列為 15、25、50 和 75 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

VIII. 封包大小吞吐量（throughput）

表 6 - 18 封包大小吞吐量

Packet size	128 bytes	256 bytes	512 bytes	1024 bytes
GGVR	Average rate: 86207.1 Peak rate: 154816	Average rate: 82621.2 Peak rate: 268128	Average rate: 82621.2 Peak rate: 268128	Average rate: 95141.6 Peak rate: 283744
GAR	Average rate: 66755.5 Peak rate: 103776	Average rate: 73213.7 Peak rate: 204288	Average rate: 73213.7 Peak rate: 204288	Average rate: 79579.8 Peak rate: 254080
AODV	Average rate: 81691.2 Peak rate: 132480	Average rate: 95915.8 Peak rate: 144704	Average rate: 95915.8 Peak rate: 144704	Average rate: 103512.5 Peak rate: 183744

GGVR 的平均吞吐量在封包大小的變化下，在封包大小為 128 bytes 時表現狀況良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。GGVR 的最高吞吐量在封包大小的變化下，在封包大小為 128 bytes、256 bytes、512 bytes 和 1024 bytes 時，表現狀況良好。尤其在封包大小為 256 bytes 和 512 bytes 時，表現顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。

IX. 傳輸量吞吐量（throughput）

表 6 - 19 傳輸量吞吐量

Rate	0.2 mb	0.4 mb	0.6 mb	0.8 mb	1.0 mb
GGVR	Average rate: 82621.2 Peak rate: 268128	Average rate: 100574.3 Peak rate: 222096	Average rate: 103284.6 Peak rate: 252320	Average rate: 102263.8 Peak rate: 200448	Average rate: 107974.0 Peak rate: 233856
GAR	Average rate: 73213.7 Peak rate: 294288	Average rate: 70269.8 Peak rate: 309024	Average rate: 75309.6 Peak rate: 339248	Average rate: 66248.0 Peak rate: 317376	Average rate: 66231.7 Peak rate: 283968
AODV	Average rate: 95915.8 Peak rate: 144704	Average rate: 108534.8 Peak rate: 200448	Average rate: 97215.4 Peak rate: 192096	Average rate: 87914.9 Peak rate: 175392	Average rate: 86222.9 Peak rate: 183744

GGVR 的平均吞吐量在傳輸量的變化下，在傳輸量為 0.6 mb、0.8 mb 和 1.0 mb 時表現良好。尤其在傳輸量為 0.8 mb 和 1.0 mb 時，表現狀況顯著良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。GGVR 的最高吞吐量在傳輸量的變化下，表現狀況穩定良好，可能是有效迴避空隙的路由方式，在此時發揮了效果。