國立交通大學

資訊科學與工程研究所

碩士論文

匿名網路之架構與金鑰管理

Architecture and Key Management of Anonymous Networks

研究生：張志誠

指導教授：楊　武　教授

中華民國　九十六　年七月

匿 名 網 路 之 架 構 與 金 鑰 管 理

Architecture and Key Management of Anonymous Networks

研 究 生：張志誠　　Student：Chih-Cheng Chang

指導教授：楊　武　　Advisor：Wuu Yang

國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Information

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年七月

匿名網路系統之架構與金鑰管理

學生：張志誠　　指導教授：楊　武 博士

國立交通大學資訊科學與工程研究所

摘要

在現代的生活中，網路幾乎是不可或缺的一部分，許多傳統的工作或娛樂以及一般日常生活行爲，都越來越仰賴網路科技，因此，對於網路的安全性，也越來越受到重視。在目前的安全技術中，針對網路傳輸封包的內容已經能提供相當程度的保護，但是仍然有其他的威脅存在。

在目前的網路架構下，即使不知道傳輸封包的內容，只要查探封包中一些不可隱藏的資訊，仍然會透露出重要的訊息。藉由封包中註明的來源和目的網路位置，可能會洩漏出通訊雙方的真實身份，因而造成隱私洩漏的危險。

在這篇論文中，我們提出一個匿名網路系統的設計，針對網路傳輸的隱私性加以保護，具有足夠的安全和效率，當使用者數量增加時，依然能維持良好的運作。

Architecture and Management of Anonymous Networks

Student：Chih-Cheng Chang **Advisor：Dr. Wuu Yang**

Department of Computer Science National Chiao Tung University

Abstract

In modern life, Internet is so important that almost part of many traditional work or entertainment and daily life, in general, are increasingly dependent on it. Hence, network security has been more and more important. In current security technology, network transmission against the contents of the packet can already provide a considerable degree of protection, but there are still exists other threats.

In current network architecture, even if the contents of packet are not known by others, the progress of transmission is still leaked out some information, as long as searching for the information that can't be hidden, will reveal important message. Observing the packet header, the source and destination of Internet location, it may leak out the identity of both parties in communication. Privacy leakage may result in dangerous.

In this paper, we represent an anonymous network system, it provides protection of transmission privacy, keeps adequate safety and efficiency, when the scale increases, still maintain well operation.

This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under the grants NSC 95-2218-E-001-001, and NSC95-2218-E-011-015.

致謝

這篇論文能完成，首先最重要要感謝指導教授楊武教授，在研究過程中遇到阻礙而進度停滯時，能適度的鞭策並且給予指引，讓我能堅持下去。老師耐心的引導，以及針對研究內容的不足或是闕漏給予意見，更是此篇論文能不斷改進的重要因素。此外，老師對於知識的好學和針對問題的嚴謹思考態度，更是值得學習的典範。

這篇論文的研究，也要感謝另一位一同合作的好夥伴冠志，資工本科出身的他，對於系統和許多專業知識方面，比我有更深厚的了解，總是能夠給予我幫助。而實驗室同學癸夫、文均，以及碩一的學弟和博班學長，以及系上其他好友，也同樣在研究或是日常生活中彼此互相幫忙，讓我的碩士研究生涯過得更順遂。另外，在研究進行實驗的階段，特別感謝清大資工石維寬教授實驗室，中興電機黃穎聰教授實驗室，逢甲資工許慶賢教授實驗室，中山電機黃宗傳教授實驗室，政大資科王乃昕學長，中央資工陳炯廷學弟，以及東華資工雍忠教授實驗室，慷慨出借主機和網路讓我們擺放網路程式進行測試。

最後，最重要的是我的家人——慈愛、開明的父母以及關心、愛護我的兄長，有他們的支持和鼓勵，從小到大才能無後顧之憂的在求學途中不斷邁進，在碩士班階段不用擔心經濟壓力或是其他問題，而能專心於論文的研究，最後終於順利完成碩士學位。僅以此文獻上由衷的感謝！

感謝國科會資通安全人才培育計畫(II)-資通安全人才培育計畫(II)之計畫編號：NSC95-2218-E-001-001 與資通安全人才培育計畫(II)-資通安全研究與教學中心(II)之計畫編號：NSC95-2218-E-011-015 補助。

目錄

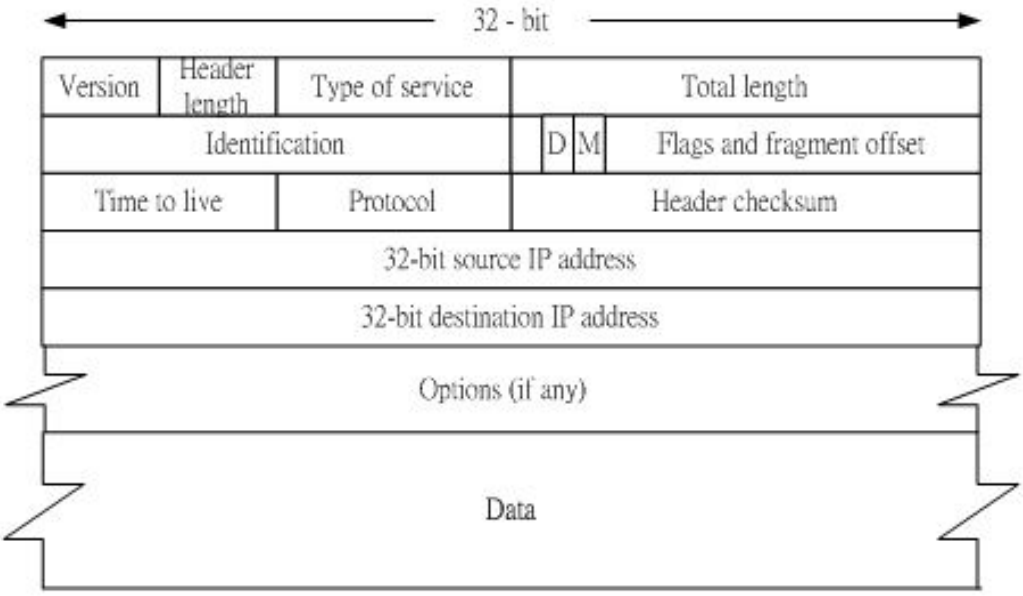
圖表目錄

第一章 緒論

1.1 背景與動機

在現代的生活中，網路幾乎是不可或缺的一部分，許多傳統的工作或娛樂以及一般日常生活行爲，都越來越仰賴網路科技。藉由網路傳輸資料的機會越來越多，因此對於網路安全性的要求，也越來越受到重視。在目前的安全技術中，針對網路傳輸封包的內容已經能提供相當程度的保護，但是仍然存在其他安全性的威脅值得重視。

在目前的網路架構下，傳輸的 IP 封包中必須包含來源及目的端的網路位置，即使不知道傳輸封包的內容，仍然會透露出重要的訊息。藉由封包中註明的來源和目的網路位置，再配合其他取得的資訊，可能會洩漏出通訊雙方的真實身份，因而造成隱私洩漏的危險。



】圖 1-1 IP封包格式[1]

圖 1-1 爲目前網路架構中 IP 封包的格式，Data 部份爲使用者傳輸的資料，其餘部份爲 IP Header，包含封包傳送的相關資訊，其中 source IP address 以及 destination IP address 分別表示此封包的來源和目的 IP 位置。因爲 Header 部份無法進行加密保護，否則封包沒辦法在網路上傳輸，因此，攻擊者只要攔截封包並讀取其中 Header 部份資訊後，即可輕易得知此封包的來源和目的。

在國防安全方面，敵國的網路駭客可透過攔截封包的方式，針對國防部的通訊對象進行觀察，若是觀察出在短期時間內，國防部與總統府或某些重要國防相關單位之間進行大量的網路通訊，即使無法得知確切的內容爲何，但可猜測出近期內可能即將進行一些重要的行動，並配合其餘情資蒐集，而洩漏出國防安全上的機密。在個人隱私方面，網路駭客可針對某個特定對象觀察其網路通訊情形，而觀察得知一些個人隱密的資訊：例如某人患有較爲隱私的疾病，時常透過相關網站了解疾病資訊，此行爲就有可能被有心人觀察出而得知其患病情形。若能設計出一套良好的匿名網路通訊系統，對於網路的安全和隱私確實能夠帶來許多的幫助，也相當具有實用的價值。

1.2 研究目標

我們的研究延續之前的研究成果[1][2][3]，匿名傳輸的方法是採用相似的概念，進而設計出一個更完整的匿名網路系統，除了針對網路傳輸的隱私性加以保護，具有足夠的安全和效率，能夠針對大多數現有的網路服務使用，當使用者數量增加時，依然能有組織地維持良好的運作。要達到匿名通訊的功能，主要的方式就是要保護網路傳輸時通訊實體的身份，我們希望能在提供足夠程度的匿名和安全性的前提下，如何使得匿名網路通訊系統的效能提高、資源浪費減少，這也是我們希望能夠達成的目標。

水能載舟亦能覆舟，匿名網路通訊系統在正當的使用上，能夠確保網路通訊的隱私和安全，但若是有心人士藉此來從事網路犯罪行爲，將會造成網安人員追蹤和查緝的困難。爲了避免匿名網路通訊系統遭致惡意的濫用，我們也設計出一套追蹤管理機制，在必要的情形下，重建出所欲追查的通訊路徑，使得犯罪的行爲能夠被追蹤出，降低匿名網路系統遭致濫用而造成的傷害。

這篇論文主要描述系統整體的架構，節點的組織和管理，以及加密金鑰分配等前置性的工作，而真正匿名傳輸以及路徑追蹤機制部份，在"匿名網路之管理與連線分析"[4]這篇論文中會有詳細的描述。

1.3 論文架構

本章針對匿名網路的應用和背景做簡單介紹，並且說明我們的研究目標和動機。第二章中提供相關的知識技術的簡介，以及現有的匿名網路系統介紹。第三章描述我們的系統設計，以及部份的實作細節。第四章介紹我們的研究成果，實作出的運作情形和實驗數據，分析本系統的安全性以及與其他現有系統的比較。第五章針對目前的研究結果做結論，以及提出一些尚待改進的目標。

第二章 相關研究

2.1 網路安全簡介

要設計良好的網路安全系統，首先要先了解有哪些攻擊的方式，以此做爲假想的攻擊，設計對應的策略。常見的攻擊模式，可以分爲下面幾種[5]：

流量分析 (Traffic Analysis)：

此種攻擊方式是藉由攔截分析網路上傳輸的封包來達成。在目前的網路架構下，封包的一些欄位無法加密，所以像是封包的大小，傳輸的數量，以及傳輸的來源和目的端都無法保密，因此很容易被攻擊者得知攻擊目標的網路活動狀態，以及網路通訊的對象等等資訊。

被動監聽 (Passive Eavesdropping)：

攻擊者攔截網路上傳輸的封包後，利用本身的破解密碼技術，而取得封包內容的資訊。

主動監聽 (Active Eavesdropping)：

攻擊者攔截網路封包後，除了破解密碼取得封包內容的資訊外，並且加以修改封包內容，使得此修改過後的封包不會被偵測出遭到惡意修改，而達成攻擊的目的，甚至可能修改封包的來源或目的端，達成 IP Spoofing 的效果。

非授權存取 (Unauthorized Access)：

這種攻擊是針對網路上必須要經過認證才能存取的 Server 來攻擊，像是無線網路的環境中，AP 可以設定成不開放給一般人存取，必須通過認證的機制才

能存取。攻擊者利用攔截封包破解密碼的方式取得相關的資訊，而達成非法的認證存取。

中間人攻擊 (Man-in-the-Middle)：

攻擊者在目標和通訊對象進行封包傳輸時，藉由攔截兩邊的封包，並且加以修改內容，使得目標和通訊對象無法得知中間的傳輸過程已經遭第三人竊取，甚至彼此傳輸的資訊遭到惡意修改。

連線劫奪 (Session High-Jacking)：

攻擊者在使用者與伺服器端完成認證後，中斷其連線，冒充取代使用者與伺服器端進行連結，由於使用者已完成認證，故攻擊者可以輕易地使用屬於這個合法使用者權限的任何資源。

重新發送 (Replay)：

攻擊者截取網路封包，然後再重新發送給接收端，例如重新發送認證訊息以假冒合法的使用者，或是透過大量地重複傳送封包使接收端癱瘓而無法正常運作。

阻斷式服務 (DoS, Denial of Service)：

這種攻擊方式，是相當難以預防，而且傷害也非常大的一種攻擊。攻擊者只要惡意在短時間內透過大量的 Service Request，使得 Server 端無法正常處理，甚至因此而潰敗，就能成功阻斷此系統的運作。

在我們的匿名網路系統中，主要是針對流量分析攻擊法來預防，希望能隱藏網路封包的真實來源和目的，提供網路通訊中的隱私，並且使用現有的加密技術

來針對其他攻擊做預防。

2.2 加解密技術

傳統的密碼系統，只有傳送端和接收端知道加解密的方式，此種方式看似安全，但因加解密碼的方式沒有經過大量的檢驗，或許本身就存在弱點，容易破解。而且在網路環境中，必須面對大量使用者，彼此之間要分配加解密的方式相當不容易。

現代密碼系統則是完全公開，除了傳送端和接收端擁有的加解密鑰不公開。密碼系統可經過公開的大眾檢視，確定其安全性，且公開的方式也容易在多方使用的網路環境中應用。目前的加解密碼系統可分成兩大類：

對稱式金鑰密碼系統 (Symmetric Key Cryptosystem)：

如圖 2-1 所示，此系統也稱密鑰加密系統，傳送端和接收端所使用的加解密金鑰相同，稱爲密鑰。傳送端將明文資料傳輸之前，先使用此密鑰加密後取得密文，再將密文傳送給接收端，接收端使用同一把密鑰解密後取得傳送端所要傳送的明文資料。

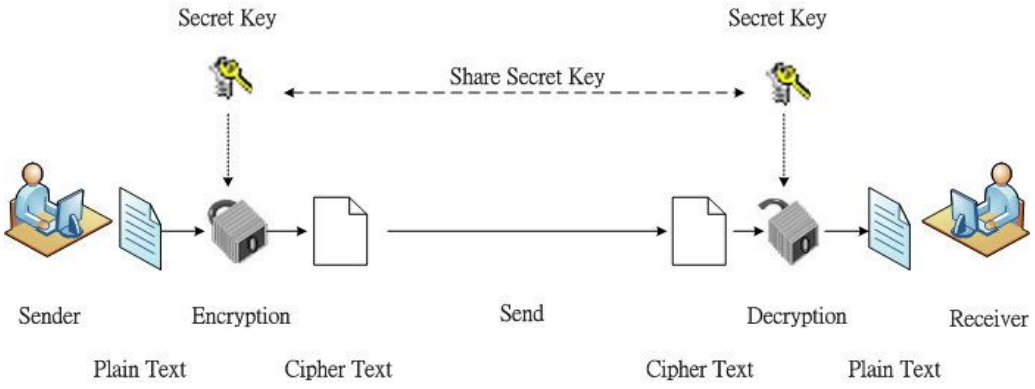


圖 2-1 對稱式金鑰密碼系統應用示例

非對稱金鑰密碼系統 (Asymmetric Key Cryptosystem)：

如圖 2-2 所示，此系統又稱公開金鑰加密系統，傳送端和接收端所使用的加解密金鑰不同，一般加密的金鑰稱作公鑰 (Public Key)，解密的金鑰稱作私鑰 (Private Key)。通訊雙方預先產生好一組公鑰和私鑰 Key Pair，將公鑰傳送給對方。傳送端傳送明文資料時，使用接收端的公鑰加密產生密文，將密文傳送到接收端。接收端收到密文，使用本身的私鑰加以解密取得明文資料。在其他的應用中，以傳送端的私鑰加密明文產生密文，將其傳送給其他擁有傳送端公鑰的接收端，而接收端以傳送端的公鑰解密取得傳送端所傳送的明文。

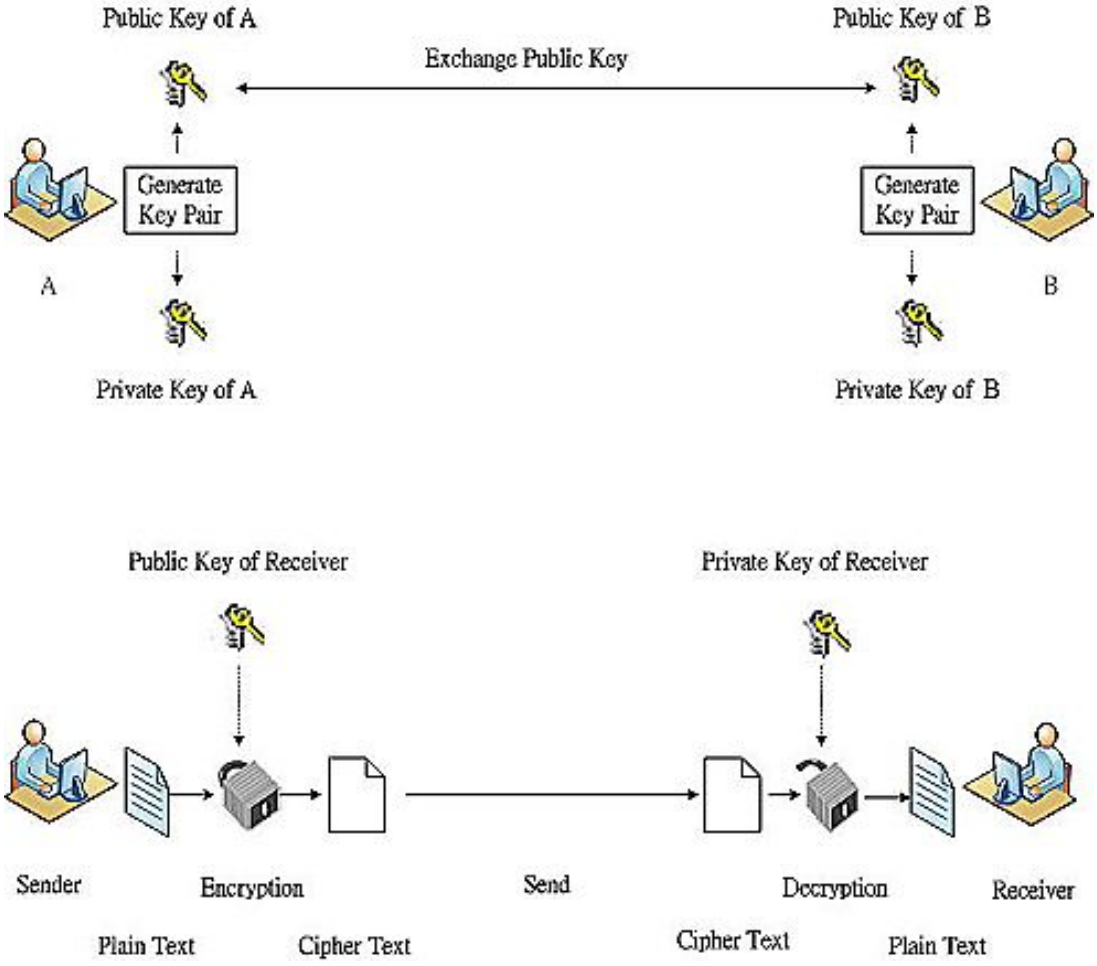


圖 2-2 非對稱式金鑰密碼系統應用示例

目前密碼系統實際應用中，大都採用前述兩種混合的方式，亦即同時使用對稱金鑰密碼系統和非對稱金鑰密碼系統。採用兩種混合的原因，主要在於這兩種密碼系統不同的特性：對稱式金鑰密碼系統加解密速度較快，但密鑰本身的配置在網路環境中是個問題，可能在過程中遭到攻擊者的攔截；非對稱金鑰密碼系統對於金鑰的分配較爲安全簡易，公鑰可任意散佈出去，不需擔心攻擊者竊取，然而其運算所需的負擔較大，加解密大量資料時，就會非常沒效率。

圖 2-3 爲混合式密碼系統的應用示例。當傳送端要傳送資料前，先從接收端取得其加密公鑰，之後傳送端本身產生一把密鑰，一般稱爲 Session Key，將此密鑰經過接收端公鑰加密後，和利用此 Session Key 加密明文資料後所產生的密文一同傳送給接收端。接收端取得此加密過後的 Session Key 和密文，先利用本身的私鑰解密，取得 Session Key，再利用 Session Key 解開密文，取得傳送端所傳送的真正明文資料，如此即可有效率且安全地進行網路傳輸。

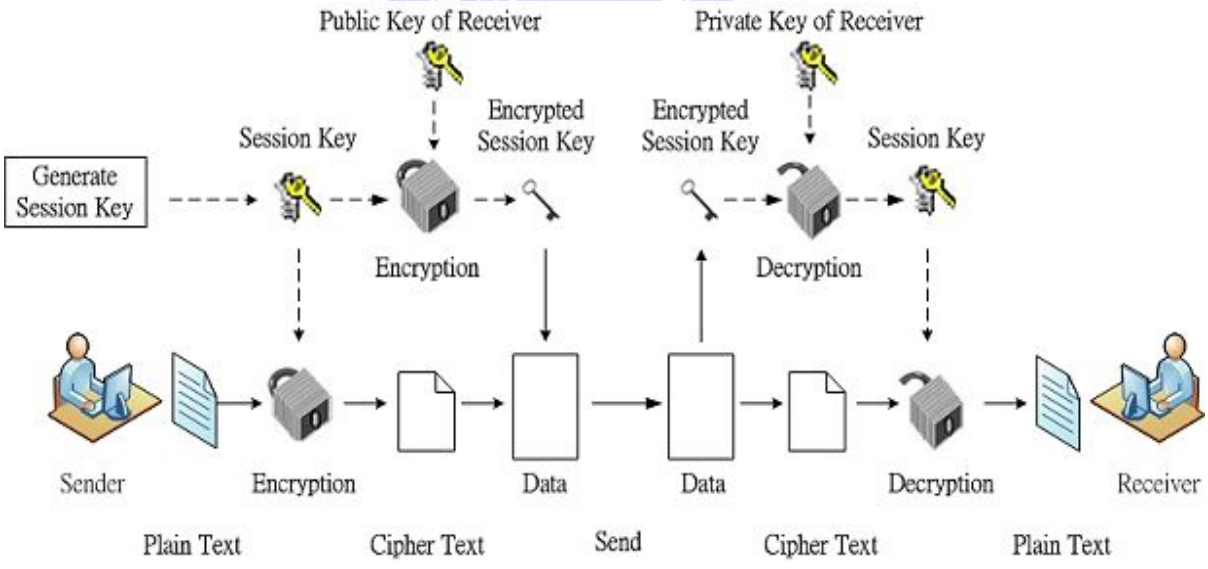


圖 2-3 混合式金鑰密碼系統應用示例

除上述介紹的應用外，也有利用公鑰私鑰特性而當作認證的數位簽章應用方式，在目前依賴網路傳輸資訊越來越頻繁的社會中，針對重要性文件，例如政府的公文或是商業上的交易文件，先利用 Hash Function，將文件取出 Message Digest，再利用本身的私鑰將其加密成爲數位簽章。將此文件與簽章傳送給擁有本身公鑰的對象，接收端則可利用相同的 Hash Function 由文件計算出 Message Digest，與利用傳送端公鑰解開數位簽章所取得的 Message Digest 做比對，若兩者相同，即可確認此文件是由傳送端所發出。此類應用因牽扯到認證的問題，所以對於公鑰的管理更爲嚴謹，需要有公正的第三方憑證管理中心 CA 來認證分佈管理各方的公鑰。

在我們的匿名網路系統中，也是採用混合式密碼系統，公開金鑰密碼系統主要用來做節點間的資訊交換，未使用金鑰認證方面的機制，所以對於公鑰的管理和分配直接在使用者節點間進行；而密鑰密碼系統則是在節點之間轉傳資料封包時用來對封包進行加解密的運作，保障匿名傳輸的匿名和安全性。

2.3 SOCKS Protocol

SOCKS 是 SOCKetS 的縮寫，是一種網路協定，主要的功用是在 Client 端與 Server 端之間進行傳輸時，可經由第三方做爲中間的傳遞者，此傳遞者亦即一般稱爲 SOCKS Server 的網路主機，可以實現許多的網路應用。如圖 2-4，可以管制內部網路與外部網路的連線，或是藉由管制外部網路的連線保護內部網路達成防火牆的功效。如圖 2-5，可以在內部網路受到一般防火牆限制某些類型或目的端的連線時，藉由 SOCKS Server 當作跳板連上本來的目的端。圖 2-6 描述 SOCKS 運作的訊息流程。

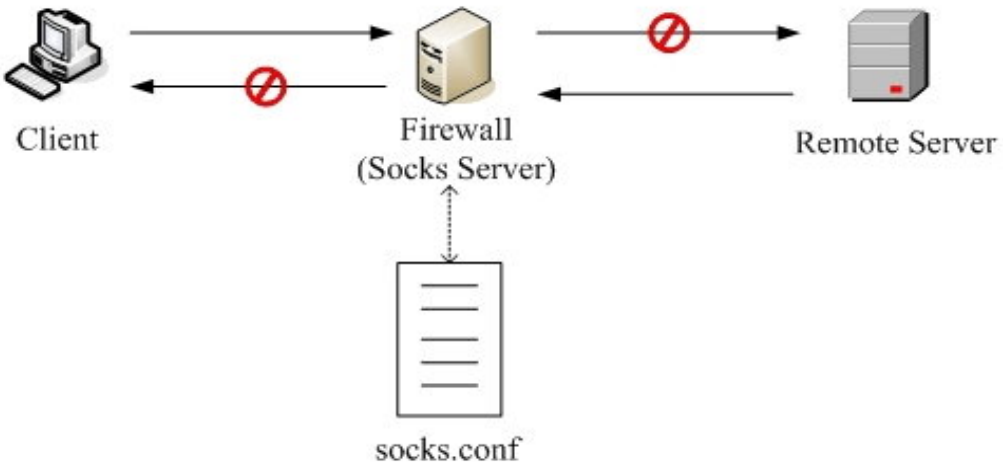


圖 2-4 SOCKS 應用範例 1

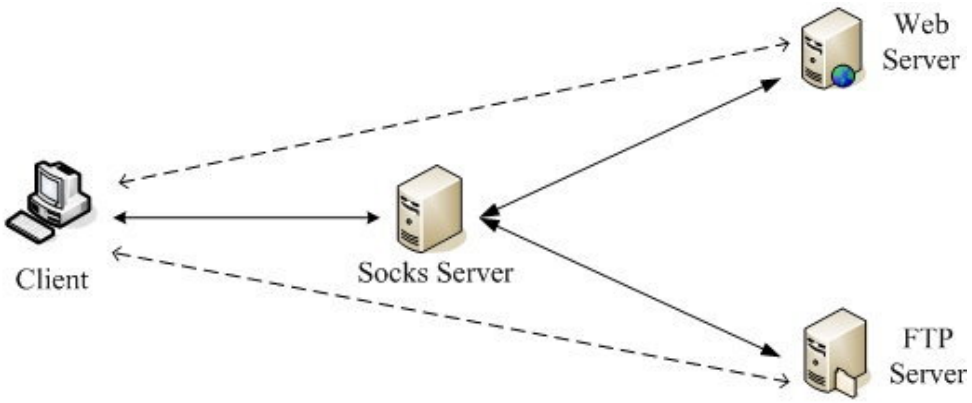


圖 2-5 SOCKS 應用範例 2

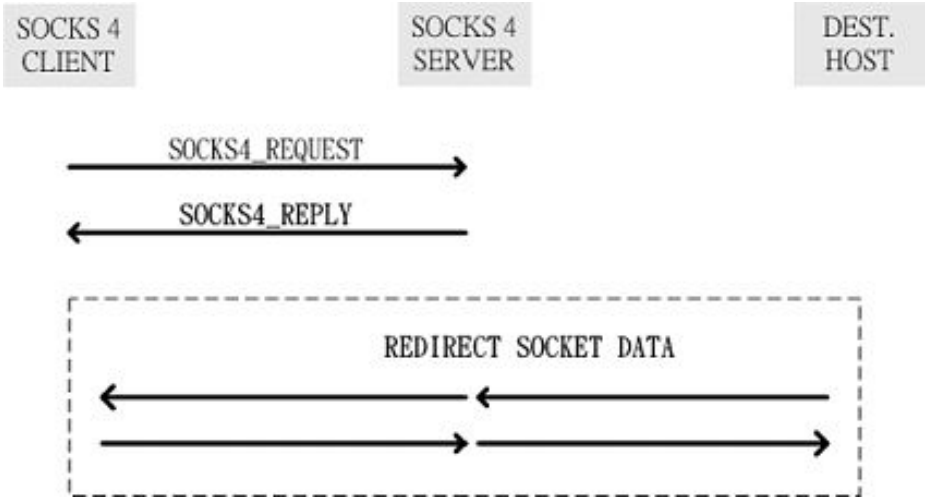
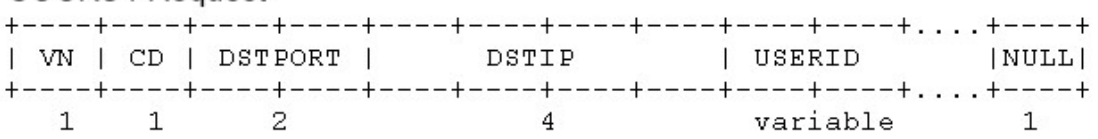


圖 2-6 SOCKS 訊息流程

SOCKS4 Request



SOCKS4 Reply

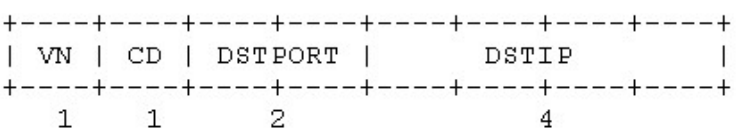


圖 2-7 SOCKS Protocol 封包

圖 2-7 是 SOCKS Protocol 的封包，其中 VN 代表 Protocol 所用版本，CD 是 Command 的縮寫，用來表示 Connect 或是 Bind 兩種 Operation，DSTPORT 和 DSTIP 分別爲目的 Port 和 IP Address，USERID 爲可變項目，有需要才設置。

我們的匿名網路系統會利用標準的 SOCKS Protocol，將網路應用程式所傳輸的封包透過我們的應用端程式，一個加入匿名處理機制的 SOCKS Server，再將封包傳送出去。因爲現有的許多網路應用程式都有支援 SOCKS Protocol，所以我們的系統可以很容易地與其他應用程式結合，若程式本身不支援，則可透過第三方軟體來達成。

2.4 匿名網路系統介紹

匿名網路系統大致上可以分成幾種類型[7]：以架構來說，有Fixed Servers和Peer-to-Peer兩種類型；以服務類型來分，可以分成High-Latency和Low-Latency兩種；以傳輸方式來分，可分成Single Proxy和Multi-Hop兩種。

Fixed Servers 類型，由系統提供者維護一群固定的 Servers，當使用者向系統

提出匿名傳輸的要求時，由這些 Server 決定使用者的封包繞徑方式，或是由這些 Server 提供使用者可用來做爲傳輸繞徑的節點資訊，再由使用者自行決定路徑。Peer-to-Peer 系統則不需中央的 Server，完全由使用者主動和其他使用者請求節點資訊，每個使用者同時擔負 Server 的角色，提供其他使用者系統的節點資訊，使用者從取得的節點資訊中選擇匿名傳輸的繞徑節點。Low-Latency 類型是指匿名通訊的傳輸在系統中可以很快得到回應，像是進行網頁瀏覽或是即時通訊等等，High-Latency 則是類似透過匿名網路系統發送 e-mail 的應用。Single Proxy 和 Multi-Hop 則是以繞徑的數量來區別，有些系統僅由 Server 端提供一個或一組 Proxy 做爲轉傳的中間點，而爲了更要求傳輸的匿名性，有些系統則是透過多點轉傳的方式進行匿名傳輸。

各類型的匿名網路系統有其特性和優缺點，Fixed Servers 的匿名網路系統，在於節點的管理和維護上較爲容易，使用者的負擔較低，取得服務的回應時間也較快，但是當使用者的數量增加時，會產生 Server 端爲了維持服務使得系統成本成長過快的缺陷，且固定式的 Server 容易成爲攻擊者的目標，攻擊者可專注於竊取分析 Server 的傳輸封包而遭受匿名性降低的風險，或是以 DoS 的方式攻擊，使系統潰敗。而 Peer-to-Peer 的架構方式較易避免 DoS 攻擊，當使用者數量增加時，各個使用者也同時分擔了系統的運作，因此對於系統不會有太大的影響。但 Peer-to-Peer 的使用者必須透過 Peer Discovery 的運作來和其他節點溝通，在系統初始時必須花費較多的時間才能取得服務，且其中滲入攻擊者的節點時，由於每個使用者的地位都相同，亦即此攻擊者也成爲系統中的 Server，對於系統的安全威脅也較大，此外，每位使用者在系統中必須負擔的責任相當，對於電腦運算或是網路能力較差的使用者節點而言，負擔也會相對較大。

Latency的差異是服務類型的不同，使用者可依自己的需要選擇。Mixmaster[9][10]和Mixminion[11][12]都是提供匿名傳輸e-mail的remailer匿名網路系統，若是要進行網頁瀏覽或是即時通訊，則有Fixed Server架構的Tor[13][14]或是Peer-to-Peer架構的Tarzan[15][16]。

繞徑節點的數量，在匿名性上來說，多點繞徑的方式可以提供較大的保障。若僅透過單個節點做爲中間的轉傳者，最大的問題就是此轉傳者可以很容易就得知此次匿名傳輸的來源和目的，攻擊者也很容易就可觀察出整個傳輸的真實路徑。然而多點繞徑的缺點就是傳輸的效率問題，除了透過中間節點轉傳本來就會造成的延遲外，若使用者本身的網路傳輸能力遠大於某個匿名路徑中間轉傳的節點，整體的傳輸效率就會受到此節點的影響。且爲了保障傳輸資訊的安全性和匿名性，轉傳的封包必須透過層層加解密的方式來達成，過多的轉傳節點必然會對整體的傳輸效率造成影響。若不考慮外部攻擊者，僅針對目的端希望能達成隱藏來源的效果，且希望能有良好的傳輸效率，透過單一Anonymous Proxy進行匿名傳輸或許是適當的選擇，Anonymizer [17]是提供此服務的一種系統。

第三章 系統設計與實作

3.1 簡介

我們的匿名網路系統，應用環境爲對於目前網路架構中任意的使用者，提供其進行網路通訊時，保護傳輸封包的來源或目的，達成匿名通訊的功能。主要的匿名傳輸方法延續之前的研究成果[1][2][3]，透過將封包層層加密且繞徑的方式來達成匿名通訊的效果，攻擊者若是僅能觀察到整體傳輸路徑中的部分，無法得知此次匿名傳輸實際的來源和目的，如圖 3-1 所示。

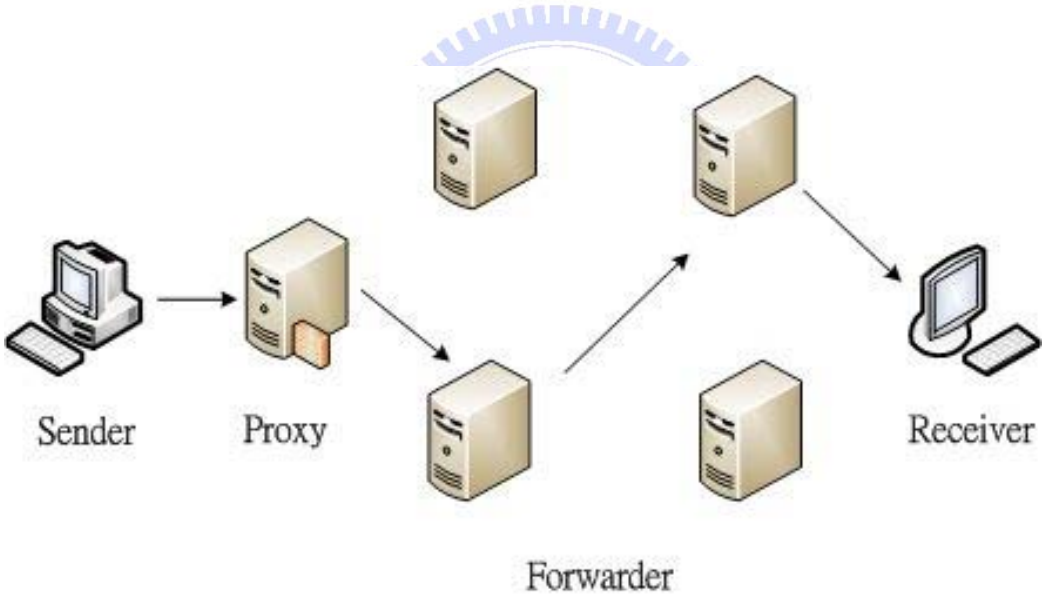


圖 3-1 匿名傳輸繞徑示例

有別於先前研究中的固定式匿名 Proxy，在我們的系統中，各使用者都有可能成爲匿名網路服務運作中的一份子。每位使用者的節點都有可能擔負轉傳的工作，攻擊者更不易觀測出傳輸的封包是否爲此節點本身所要傳送的封包，增加流量分析的困難度。由於每個節點都要負責部份的工作，要使用我們的系統，必須安裝一個應用程式，其中分成許多部份處理各個不同的工作。

我們的系統除了保有匿名性的基本功能之外，也將整個系統設計得更完整。主要加強的部份在於使用者資訊的分佈與管理、節點的組織成長，以及路徑追蹤機制的建立，避免系統遭到不肖份子使用，進行非法的網路行爲。

3.2 系統模型和假設

所有網路上的節點，我們以 IP Address 表示其身份。假設攻擊者會在系統內佈置一些惡意節點，亦即由攻擊者所掌控，但經由正常管道進入系統，以觀察系統內部網路傳輸情形進行分析。惡意節點也可能直接成爲我們匿名傳輸 Tunnel 中的節點，直接觀測傳輸的封包， 在 Tunnel 中，包含的惡意節點越多，此次匿名傳輸的真實路徑被分析出的機會越高。

此外，我們假設攻擊者所能夠觀測的Domain有一定的限制，所以在建立Tunnel的過程中，盡量選取不同Domain的節點，能帶來較高的隱私性[15]。相同domain的IP Address，會有一定長度相同的Prefix，將其以unsigned int表達時，彼此會是接近的數字，因此，我們以IP Address作爲依據，將相近的節點群聚在一定相近的群組中，若是有惡意攻擊者的節點，也能將其限制在系統中鄰近的區域，減低其感染整個系統的機會。

3.3 架構

在架構上，我們將整個系統分成三層：

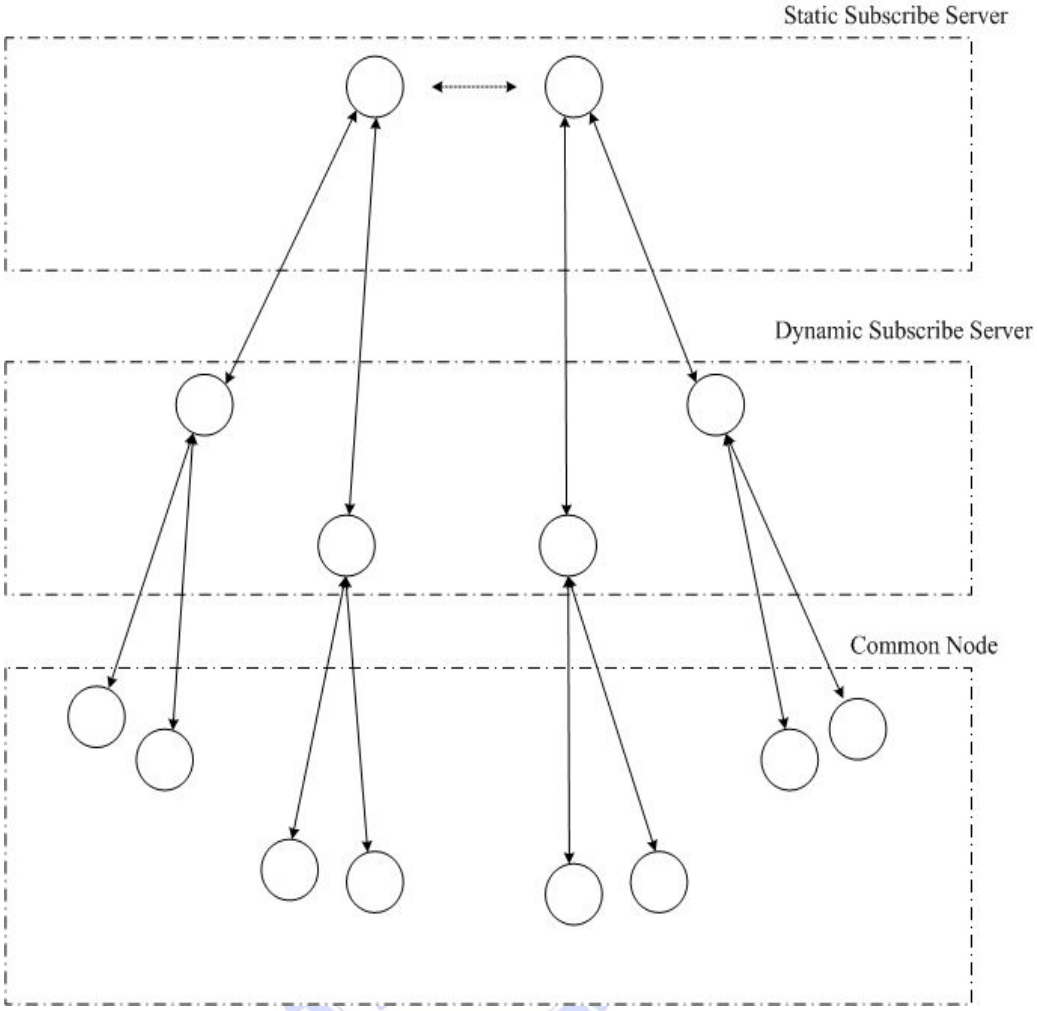


圖 3-2 系統架構

3.3.1 Static Subscribe Server

如圖 3-3，Static Subscribe Server 是由系統提供維護的 Server，負責一般使用者進入系統時的註冊，以及管理各節點的分佈組織，以及各群組之間節點數量的平衡負載，偵測各群組的 Dynamic Subscribe Server 是否仍在系統中運作。

各 Static Subscribe Server 擁有部份的節點資訊，每個 Server 的節點資訊合併即爲整個系統的節點資訊，分擔儲存和管理節點資訊的成本。各 Server 之間進行訊息交換，擁有彼此的部份訊息：管理的節點數目、群組數目以及各自管理的

Dynamic Subscribe Server，稱爲 Slave Dynamic Subscribe Server，負責將系統中所有的 Dynamic Subscribe Server 的資訊發佈給各自管理的 Slave Dynamic Subscribe Servers。

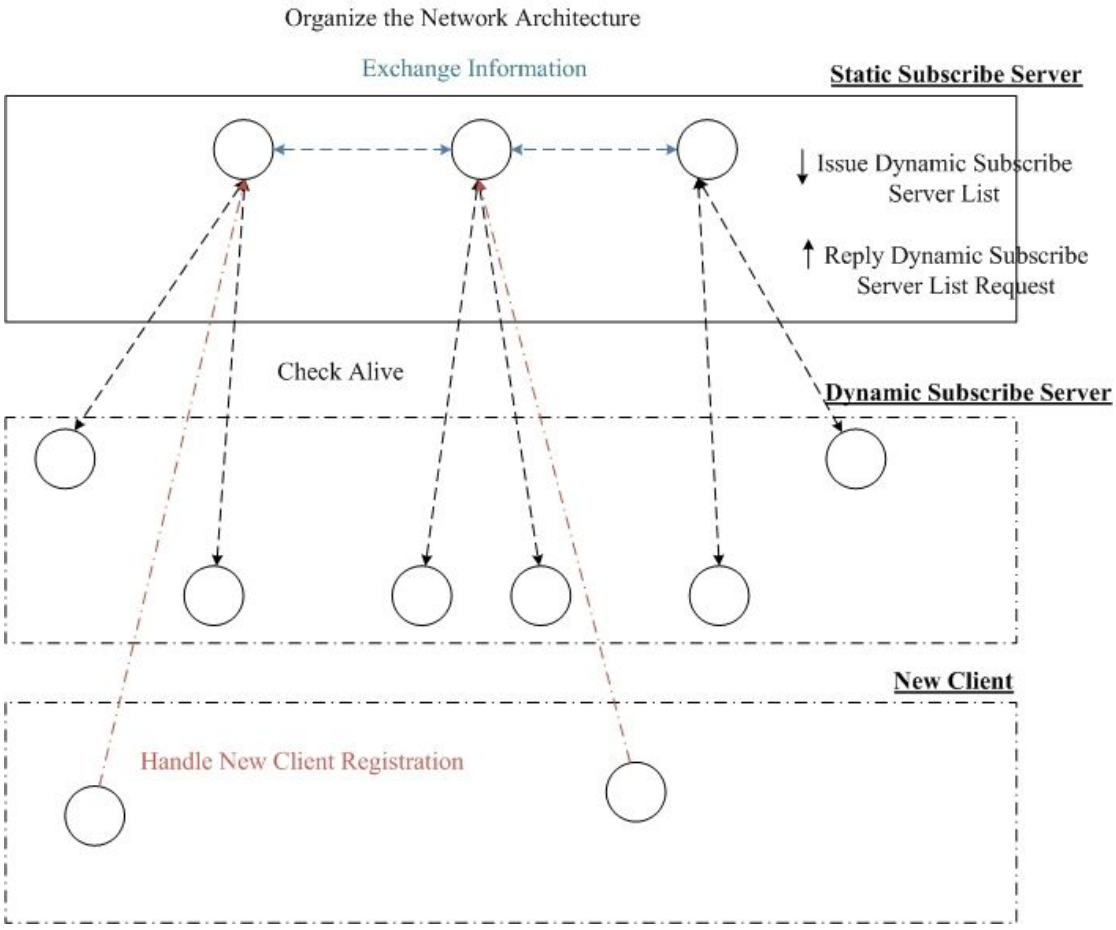


圖 3-3 Static Subscribe Server 運行機制

3.3.2 Dynamic Subscribe Server

如圖 3-4，每個 Dynamic Subscribe Server 由一個 Static Subscribe Server 管理，稱爲 Master Static Subscribe Server，擁有 Master Static Subscribe Server 所管理 Common Node 的部份資訊，這些 Common Node 即由此 Dynamic Subscribe Server 管理，稱爲 Slave Common Node，此外 Dynamic Subscribe Server 也擁有整個匿名網路系統中所有 Dynamic Subscribe Server 的資訊，或是一個可設定的上

限值數量，再動態地和 Master Static Subscribe Server 進行更新。

Dynamic Subscribe Server 和其管理的 Slave Common Node 稱爲一個群組，必須負責偵測和回報 Slave Common Node 是否仍在系統中運作，若是某 Common Node 不存在系統中，則向上回報 Master Static Subscribe Server，自己也必須週期地向 Master Static Subscribe Server 回報本身仍在系統中運作的訊息。負責傳遞 Master Static Subscribe Server 和 Slave Common Node 之間的訊息，包括從 Master Static Subscribe Server 端傳遞訊息給 Slave Common Node，以及從 Slave Common Node 端傳遞訊息給 Master Static Subscribe Server。就是當使用者向其詢問可用的匿名節點資訊時，回應其管理的 Slave Common Node 資訊，以及回應 Slave Common Node 向其詢問 Dynamic Subscribe Server 的資訊。

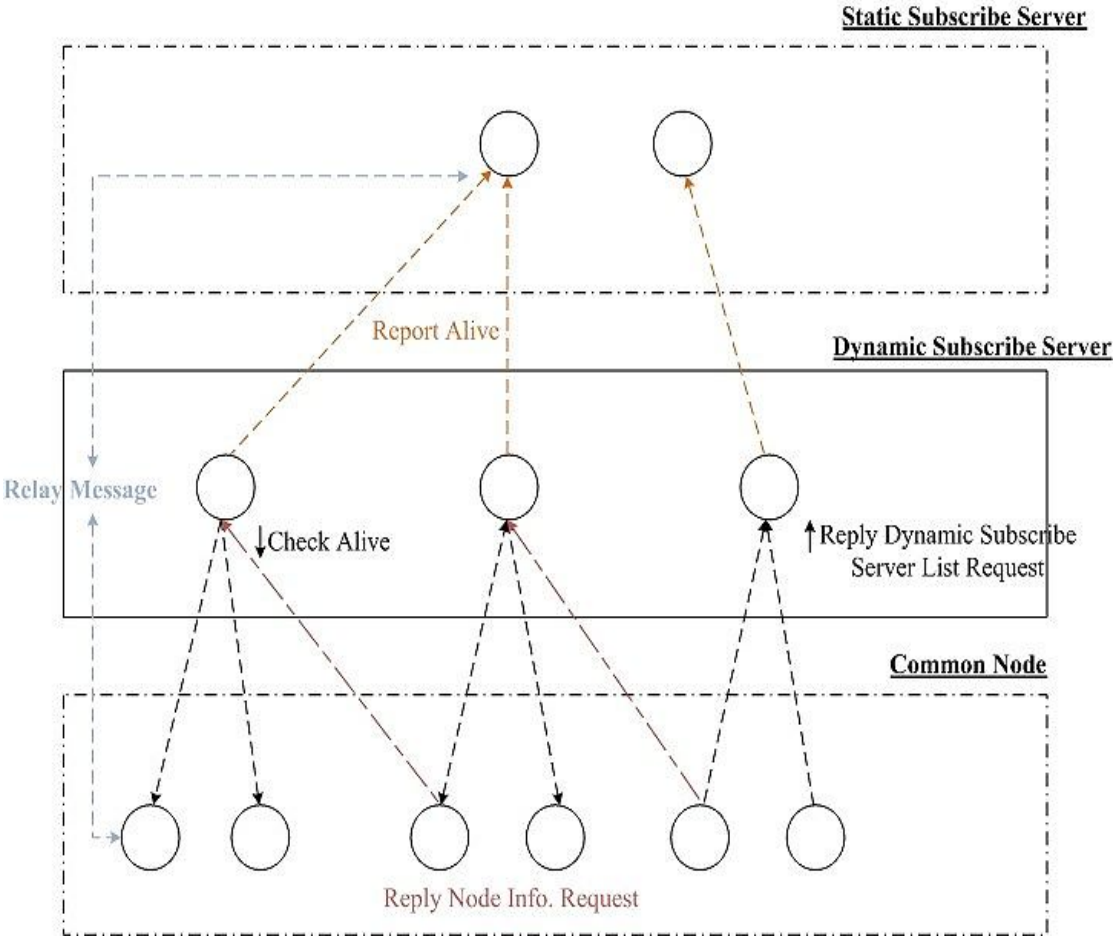


圖 3-4 Dynamic Subscribe Server 運行機制

3.3.3 Common Node

如圖 3-5，一般的匿名網路使用者，除了週期向其所屬的 Dynamic Subscribe Server，稱爲 Master Dynamic Subscribe Server，回報自己的運作狀況外，不需負擔其餘工作。Common Node 隨著使用者增加後，有可能轉變成爲 Dynamic Subscribe Server，分擔系統的運作。

當使用者進入系統後，若是恰好被選爲 Dynamic Subscribe Server，由 Static Subscribe Server 主動聯繫。否則被分配到其所屬的群組，管理該群組的 Dynamic Subscribe Server 主動向 Common Node 聯繫。一開始 Common Node 除了 Master Dynamic Subscribe Server 之外，沒有其他節點的資訊，於是向其要求所有或是一個設定上限值節點數量的 Dynamic Subscribe Server 資訊，存在 Dynamic Subscribe Server List 中，之後 Common Node 再隨機和 List 中的 Dynamic Subscribe Server 要求節點資訊。之後 Common Node 也會動態地向 Master Dynamic Subscribe Server 更新 Dynamic Subscribe Server List。

當 Common Node 取得節點資訊後，在進行匿名傳輸之前，會先由所取得的節點資訊中選取節點進行 Tunnel Setup，當此程序完成，便將 Tunnel 的資訊存到 Tunnel List 中，作爲進行匿名傳輸時的 Anonymous Tunnel。Dynamic Subscribe Server 同時也具有 Common Node 的身份，同樣也會和其他 Dynamic Subscribe Server 要求節點資訊。

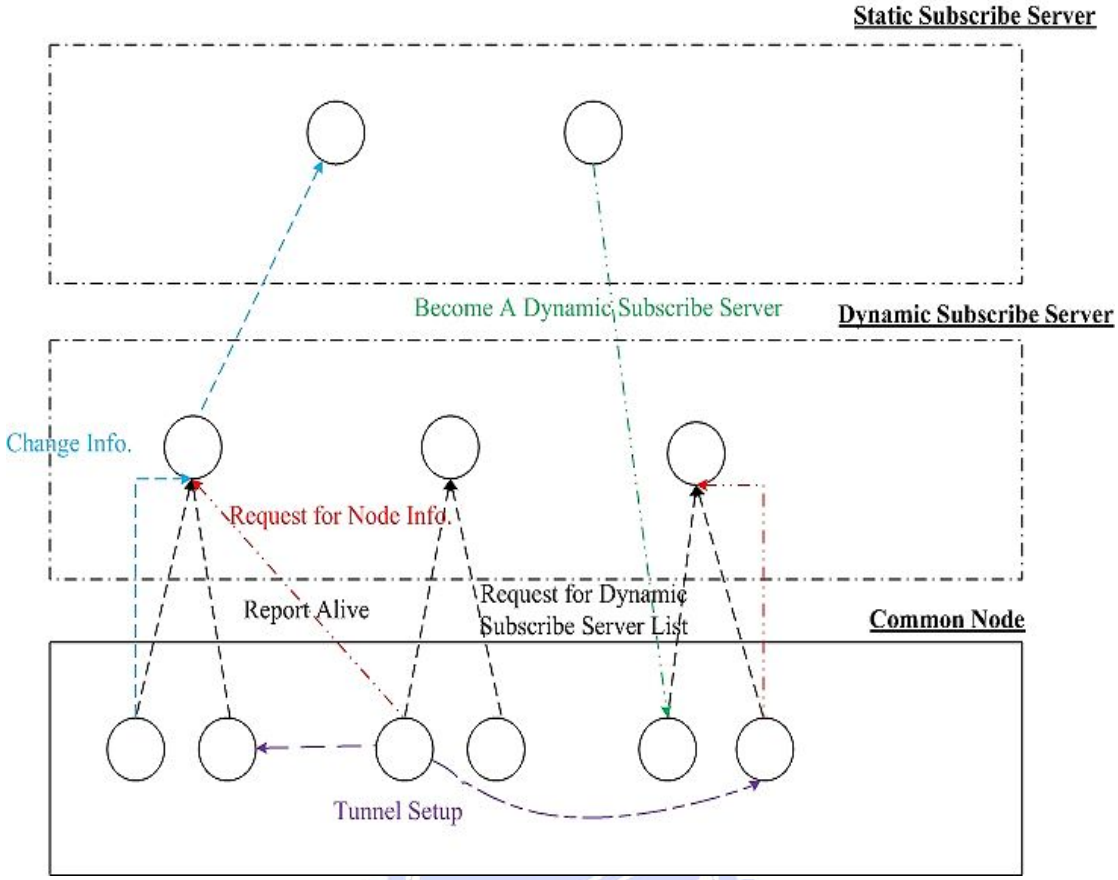


圖 3-5 Common Node 運行機制

3.4 Client 端程式模組

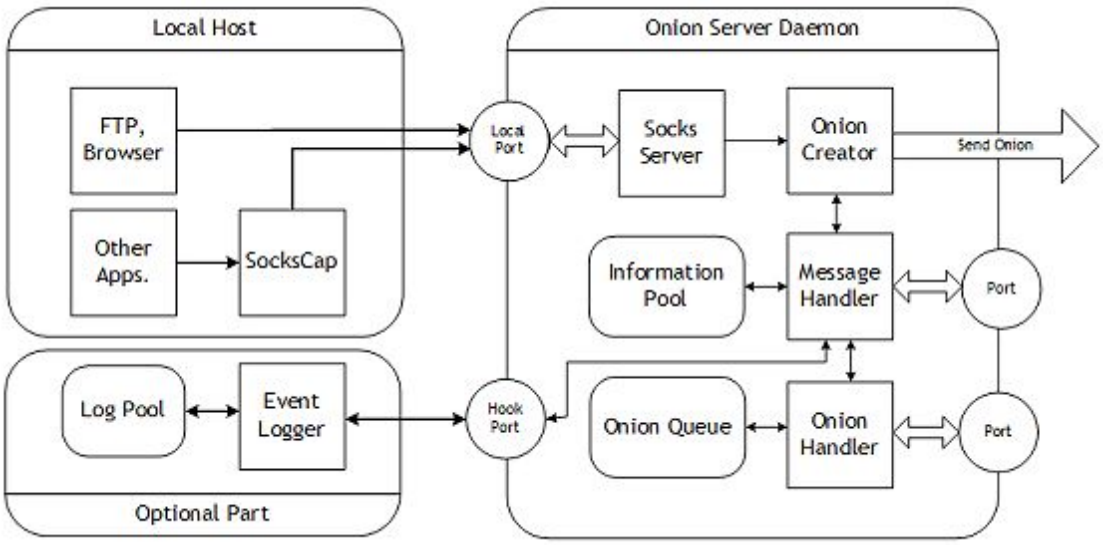


圖 3-6 Client 端程式模組

圖 3-6 爲 Client 端的程式模組圖示，其中各部份簡介如下：

Local Host

一般的 TCP 網路應用程式，如 Web Browser，FTP，即時通訊軟體，將本身的 SOCKS Proxy 模式設定成本機端位置和 SOCKS Server 所使用的 Port，則此應用程式傳輸的封包會透過本機端的 SOCKS Server 再傳輸出去。若應用程式本身不支援 SOCKS Proxy，可透過第三方軟體來將網路封包傳送到 SOCKS Server。

SOCKS Server

接收由本地端應用程式透過 SOCKS 設置傳送來的封包，一般的 SOCKS Server 會在確認 Header 之後，馬上將封包依照 Header 中標示的目的端以及 Command 模式進行轉傳或是 Bind Port 的處理，我們的 SOCKS Server 會將此封包傳送給 Onion Creator 來做匿名傳輸的前置處理。

Onion Creator

從 Message Handler 處取得先前建立好的 Tunnel 相關資訊，包括 Tunnel 中選取的節點 IP Address 和 Port，以及預先配置好的 Session Key，將封包層層加密後做成 Onion Packet，傳送到 Tunnel 中的第一個節點。

Message Handler

處理節點之間的訊息交換，Common Node 接收由 Master Dynamic Subscribe Server 所傳送的 Command Message，Dynamic Subscribe Server 接收由 Master Static Subscribe Server 傳送的 Command Message。存取 Information Pool 裡面的各式資料，如 Dynamic Subscribe Server List 或是 Node Table，以及 Tunnel Setup 完成後

建立的 Tunnel List，當別的節點要求與其建立 Tunnel 時，負責處理記錄相關資訊儲存於 Information Pool 中。

Onion Handler

接收處理別的節點傳送過來需要轉傳的 Onion Packet，經由預先配置好的 Session Key 解密，處理後再轉傳到 Tunnel 中的下一節點。或是不立即轉傳 Onion Packet，而將 Onion Packet 送到 Onion Quere 存放，等待其他的 Onion Packet 接收後再做處理。

Information Pool

儲存各種節點或 Tunnel 的相關資訊，由 Message Handler 維護，以供查詢使用或是更新內容。主要有 Dynamic Subscribe Server List，Node Table，Tunnel List 等等。

3.5 Server 端程式模組

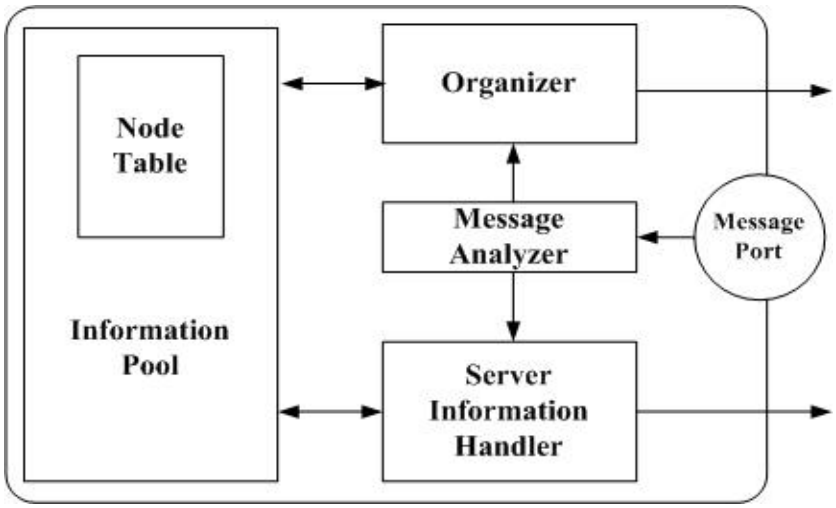


圖 3-7 Server 端程式模組

圖 3-7 爲 Server 端程式模組，其中各部份簡介如下：

Message Analyzer

接收外部傳送來的 Command Message，分析後送到其他模組執行。與其他 Static Subscribe Server 有關的指令，送到 Server Information Handler 執行，和 Slave Dynamic Subscribe Server 或 Slave Common Node 有關的指令，送到 Organizer 中處理。

Information Pool

儲存節點相關資訊，以及其他 Static Subscribe Server 的資訊。提供給其他模組使用到相關資料時查詢或更新。

Organizer

負責維護管理節點組織狀態，將結果分配給所管理的節點，要求 Slave Dynamic Subscribe Server 更新 Slave Common Node List，或是要求 Common Node 轉變成 Dynamic Subscribe Server 等等。

Server Information Handler

處理有關其他 Static Subscribe Server 的事務，更新維護彼此間的資訊，以及共享 Dynamic Subscribe Server List 等等。

Node Table

儲存 Static Subscribe Server 所管理的所有節點資訊，包括 Slave Dynamic Subscribe Server 和 Slave Common Node，依照群組關係儲存，並且在其中進行組織和分配，將結果回傳 Organizer，由其分佈給其他節點依照分配後的結果執行。

3.6 實作細節

3.6.1 使用者註冊

當使用者運行 Client 端程式後，首先要做的步驟便是將本身的資訊加入到系統中。

向系統進行註冊之前，首先 Client 端程式會先產生一些必要的資訊：一個亂數 Authentication Number，以及所採用的公開金鑰密碼演算法產生一對 Key Pair，以及利用本身的 CPU 處理速度，RAM 的大小，以及網路傳輸能力運算產生一個 Efficiency 能力值。

Authentication Number 是由本機端隨機產生的一個亂數，目前是採用 32 bit 的大小。這個亂數是用來做爲上層節點發送 Command 給下層節點時，做爲認證的憑據之用。僅採用 32 bit 是爲了要減少資料儲存的成本，而且當攻擊者多次發送不正確的 Authentication Number 後，Client 端可以回報此訊息給系統，踢除惡意攻擊者，避免暴力破解。之後運作過程中，Client 端定期更新 Authentication Number。產生的公開金鑰 Key Pair 並不在註冊的過程中傳送給 Static Subscribe Server，在之後需要交換訊息時才在節點間交換，減少 Server 端儲存 Public Key 的成本。

由 CPU，RAM，和網路傳輸能力計算得來的 Efficiency 值，會用在節點數量增加必須新增群組時，選取新 Dynamic Subscribe Server 的依據。各節點進行 Tunnel Setup 時，也可供 Client 端做爲選取節點的參考依據。

目前我們的設計，是採用 8 bit 的數字來存放，將 255 分成 CPU，RAM 和網路能力各佔 85。將 CPU 的運算能力以 100 MHZ 爲單位，每 100MHZ 記爲 3 分，若總值超過 85 即以 85 記。RAM 以 100 MB 爲單位，每 100 MB 記爲 4 分，總值超過 85 以 85 記。網路能力以 100 KB/s 爲單位，每 100 KB/s 記爲 2 分，總值超過 85 以 85 記，圖 3-8 爲計算公式。

$$\text{Efficiency} = \text{CPU Frequency (MHZ)} \times \frac{3}{100} + \text{RAM Size (MB)} \times \frac{4}{100} + \text{Transmission Data Rate (KB/s)} \times \frac{2}{100}$$

圖 3-8 Efficiency 計算公式

在使用者取得的 Client 端程式裡，包含了部份 Static Subscribe Server 的列表，程式運行後會隨機從其中選取一個做爲 Client 進行註冊的對象，再由此 Static Subscribe Server 依照 Client 的 IP Address 計算出 Client 應該歸屬的 Static Subscribe Server，將此註冊訊息轉傳過去，真正進行註冊，最後依然從原來註冊的 Static Subscribe Server 回傳註冊成功訊息給 Client，註冊訊息如圖 3-9 所示，包含 IP Address，用來進行訊息交換的 Message Port，認證用的 Authentication Number，以及前述的 Efficiency 值。

4 IP: IP Address
2 MPORT: Message Port
4 AN: Authentication Number
1 EFF: Efficiency

圖 3-9 Client 註冊訊息

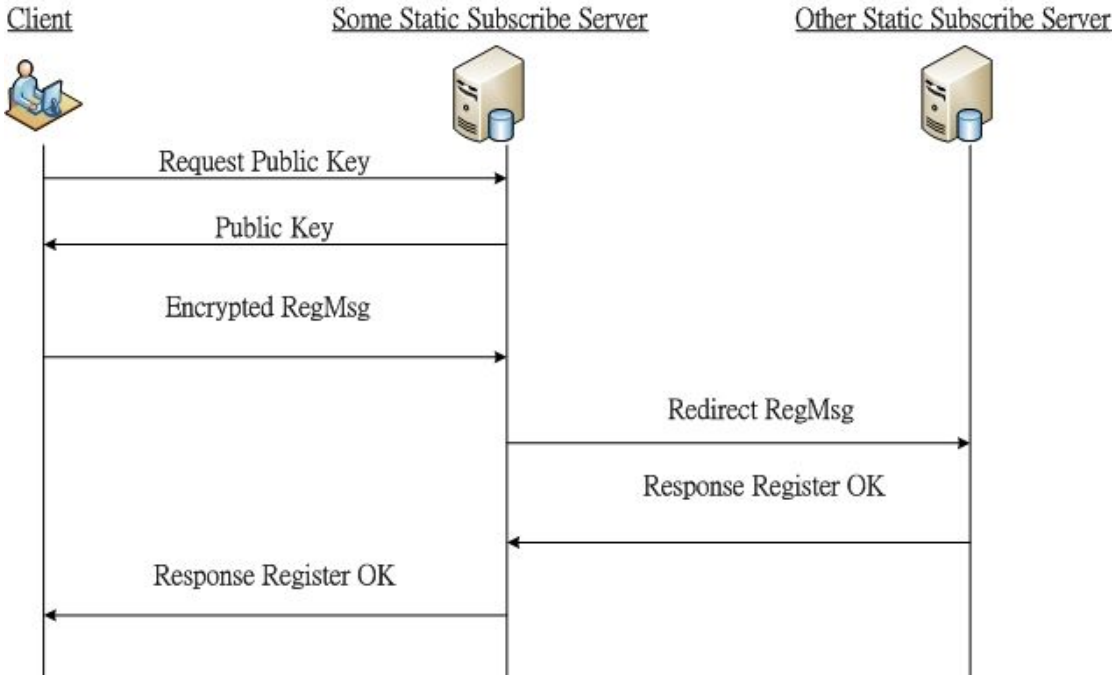


圖 3-10 使用者註冊 Message Flow

圖 3-10 爲 Client 進行註冊的簡易 Message Flow。一開始先向 Static Subscribe Server 要求 Public Key，取得後，將註冊資訊 RegMsg 用 Public Key 加密後傳送，確保註冊資料的安全。註冊過程中的轉傳註冊資訊未必會發生，因爲可能 Client 恰好就是屬於此 Static Subscribe Server 管理。

3.6.2 節點管理

節點管理主要是由 Static Subscribe Server 中的 Node Table 負責。每個 Static Subscribe Server 中的 Node Table，儲存管理的所有節點資訊，新 Client 進行註冊時，將註冊資訊加入 Node Table 之中，由 Node Table 內部進行組織和分配的運算。

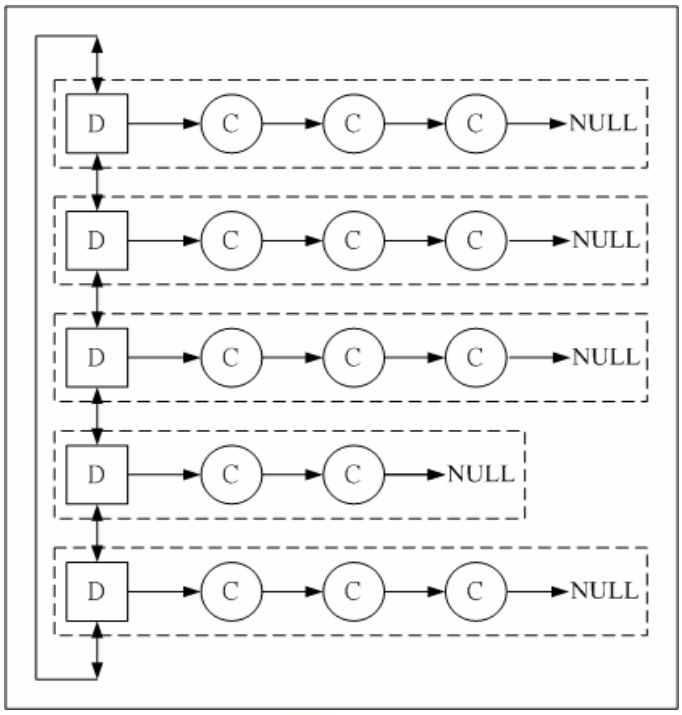


圖 3-11 Node Table 資料結構

圖 3-11 爲 Node Table 的資料結構，其中 D 代表 Dynamic Subscribe Server 節點，C 代表 Common Node 節點，虛線框內代表一個群組，由其中的 Dynamic Subscribe Server 管理。群組內的 Common Node 節點依照 IP Address 排序，而群組之間也是依照 IP Address 的順序排列。此外，爲了考量節點數量增加時，搜尋節點花費的時間成本，Node Table 內存有兩個 Search Tree 來幫助搜尋節點。

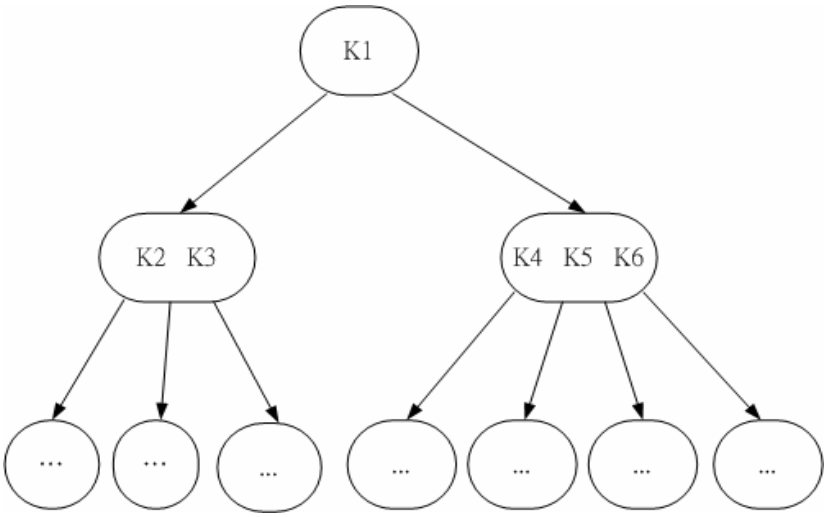


圖 3-12 B-Tree 資料結構

B-Tree of order m, may be an empty tree, or it holds the properties：

1. Root has at least two child nodes.
2. Except root and failure node, all other nodes have at least ⌈m/2⌉ child nodes.
3. Failure nodes are at the same level.
4. Search time complexity = height of search tree.
5. Most no. of nodes = $(m^h - 1) / (m - 1)$.
6. Least no. of nodes = $(2 * \lceil m/2 \rceil^{h-1} - 1) / (\lceil m/2 \rceil - 1)$
7. Most no. of key = $m^h - 1$
8. Leat no. of key = $2 * \lceil m/2 \rceil^{h-1} - 1$

表 3-1 B-Tree 的相關特性

如圖 3-12，我們採用B-Tree做爲Search Tree，從網路上取得的B-Tree Source Code[18]加以修改，表 3-1 爲B-Tree的相關特性。每個B-Tree Node以IP Address爲Key，另存放Dynamic Subscribe Server或Common Node在Node Table中節點的指標，進行搜尋時，以IP Address搜尋到B-Tree中的Key值後，再從相對應的指標找到Node Table中存放網路節點的位置。一般Search Tree找尋目標Key值，若搜尋到則回報相關資訊，找不到則搜尋失敗。我們將此Search Tree新增功能，能夠找到比Key值小且最接近的目標。在Node Table中包含以下操作：

Add：新增 Common Node 節點。

當 Client 註冊節點時，Static Subscribe Server 將節點資訊新增到 Node Table 中。由 Common Node Search Tree 搜尋到比新增節點 IP Address 小且最接近的節點位置後插入，如圖 3-13。

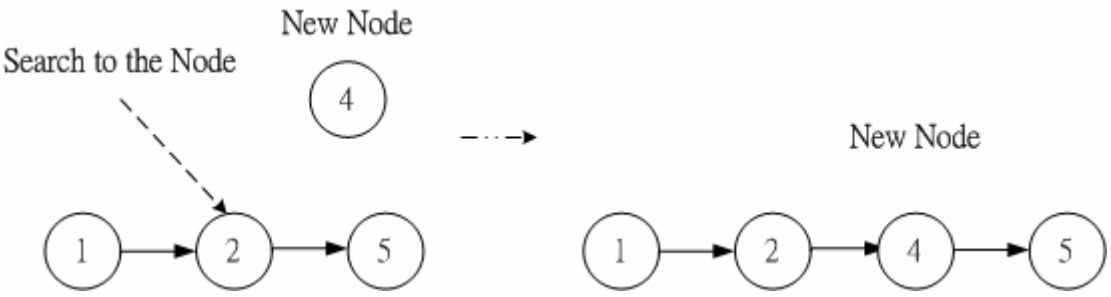


圖 3-13 Node Table: Add

Remove：移除 Common Node 節點。

當 Common Node 斷線或是停止運作時，由 Dynamic Subscribe Server 回報，將此 Common Node 的 IP Address 送入 Node Table 之中移除。同樣利用 Search Tree 搜尋到 IP Address 小且最接近的節點後，將其下一節點移除，如圖 3-14。

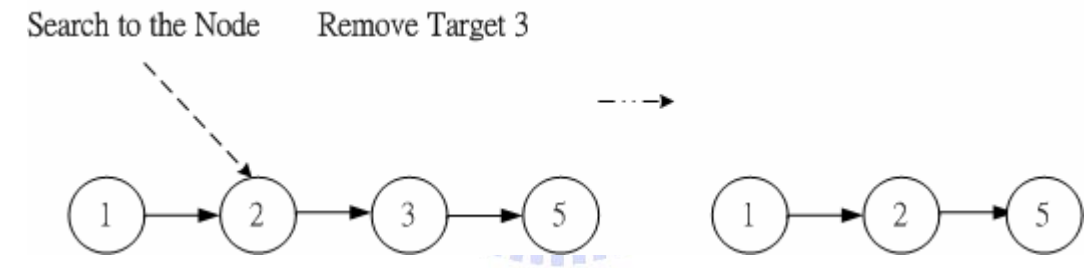


圖 3-14 Node Table: Remove

Add Group：新增群組。

當群組中的 Slave Common Node 超過系統設置的 Max Group Number 之後，從一群組的 Slave Common Node 之中選取 Efficiency 較高者，使其成爲新群組的 Dynamic Subscribe Server，將原群組中的 Slave Common Node 分成兩部份，比對原群組的 Dynamic Subscribe Server 與新 Dynamic Subscribe Server 的 IP Address 後，IP 位置較小者，管理前半部，較大者管理後半部。當群組的數目到達 Max Group Number 時，將 Max Group Number 增爲兩倍，如圖 3-15。

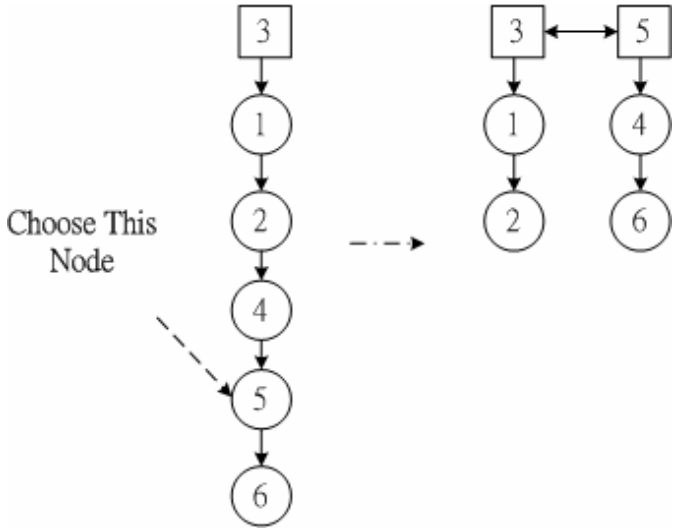


圖 3-15 Node Table: Add Group

Remove Group：移除群組。

當群組中的 Slave Common Node 爲 0 或是少於 1/4 Max Group Number，將此群組中的 Slave Common Node 分爲前後兩半，分別加入此群組中的前後群組，而 Dynamic Subscribe Server 變更回 Common Node 的身份，由 Add 加入系統中。當群組數目僅爲 Max Group Number 的 1/4 時，將 Max Group Number 減爲一半，如圖 3-16。

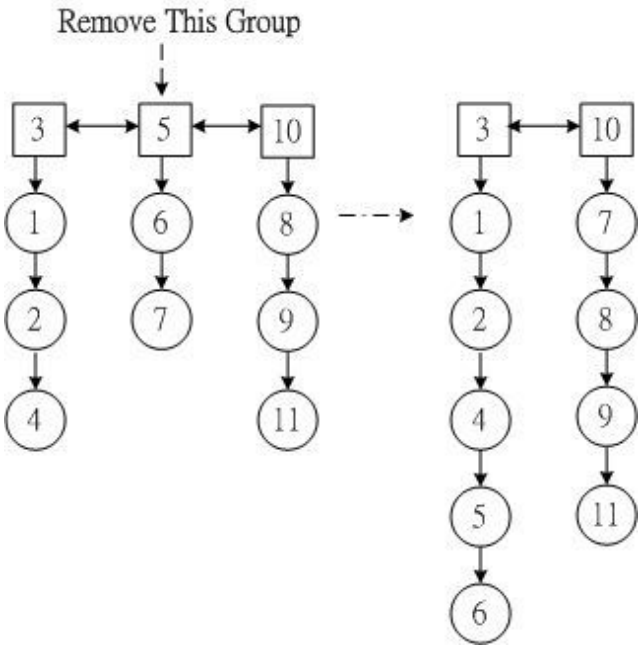


圖 3-16 Node Table: Remove Group

Balance: 平衡群組間的節點數目。

因爲 Dynamic Subscribe Server 是由一般使用者的 Common Node 之中選取，爲了避免少數使用者負擔較大，我們希望能維持群組之間節點數目的平衡，因此，在前面提及的 Add、Remove、Add Group、Remove Group 操作之後，會再進行 Balance。

首先計算 Static Subscribe Server 之中每個群組應該負責管理的 Common Node 數目平均值：

Avg. Common Node No. of Each Group = All Common Node No. / Group No.

取此平均值的 Ceiling 和 Floor，定爲每個群組管理 Common Node 數目的上下界：

Upper Bound = Ceiling (Avg. Common Node No. of Each Group)

Lower Bound = Floor (Avg. Common Node No. of Each Group)

Balance 分爲 Add Balance 和 Remove Balance 兩種模式，另分爲 Previous Balance 和 Next Balance 兩種方向。進行 Add 或 Remove Group 之後，會進行 Add Balance，而 Remove 或 Add Group 之後，會進行 Remove Balance。Add Balance 是指進行 Balance 操作的群組管理的 Common Node 數目超過 Upper Bound，將部份的節點移往鄰近的群組管理；而 Remove Balance 則是指進行 Balance 操作的群組管理的 Common Node 數目少於 Lower Bound，由鄰近群組調動過來。Direction 的選擇一般是依據進行 Balance 的群組和兩邊群組的節點數目比較的結果。進行 Add Balance 時，若是 Previous Group 的節點數目較少，就進行 Previous Balance，反之進行 Next Balance；進行 Remove Balance 時，若是 Previous Group 的節點數目較多，進行 Previous Balance，反之進行 Next Balance。

此外，爲了避免連續新增的節點落在 Node Table 中兩極處，造成群組數目的偏斜或是 Balance 的運作需要花費較多的時間，如圖 3-17，所以我們將群組之間的關係造成環狀架構，如圖 3-18。

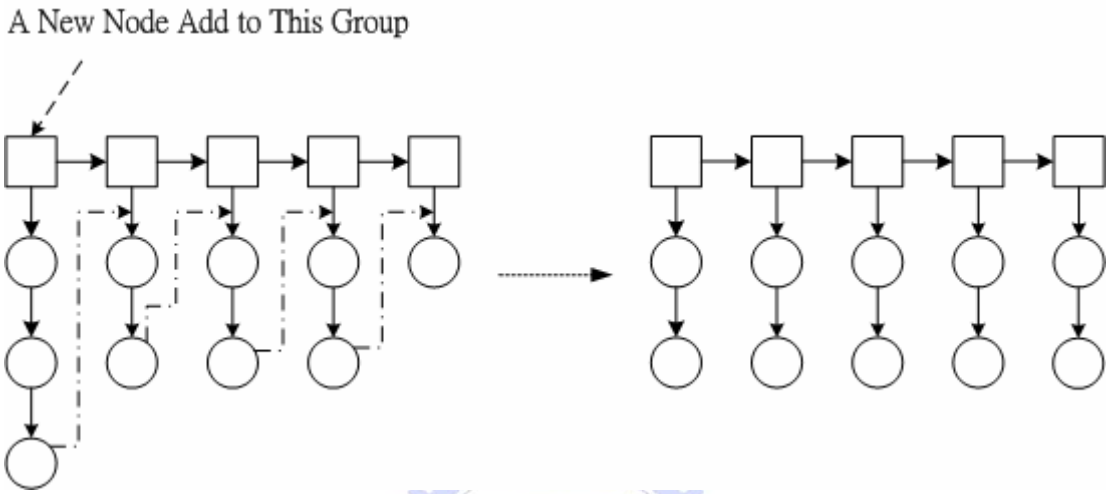


圖 3-17 線性群組架構的 Balance

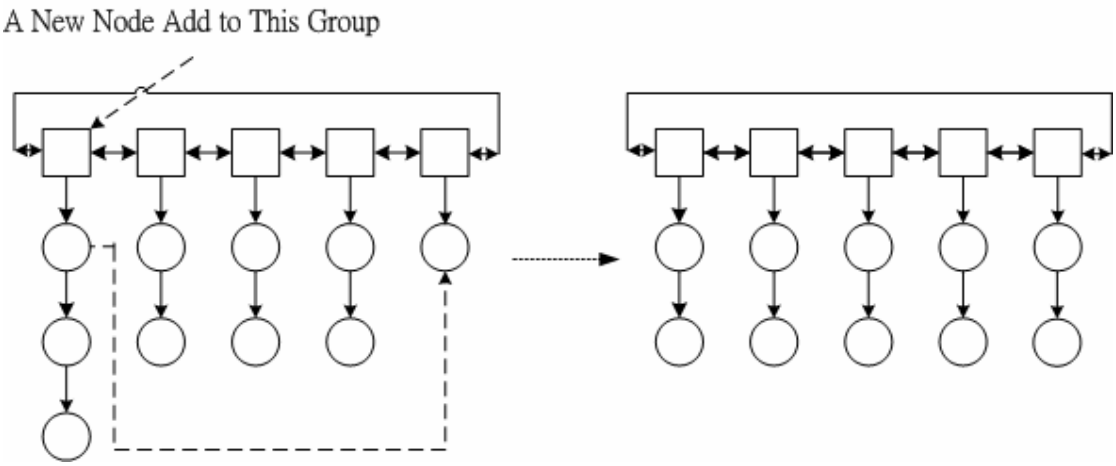


圖 3-18 環狀群組架構的 Balance

因爲環狀架構的緣故，IP Address 較大的節點和 IP Address 較小的節點可能會落在同一群組中，因此在 Balance 選擇 Direction 時，當鄰近兩側的群組擁有相同節點數目時，會傾向於讓 IP Address 分佈正常的方式，如圖 3-19。

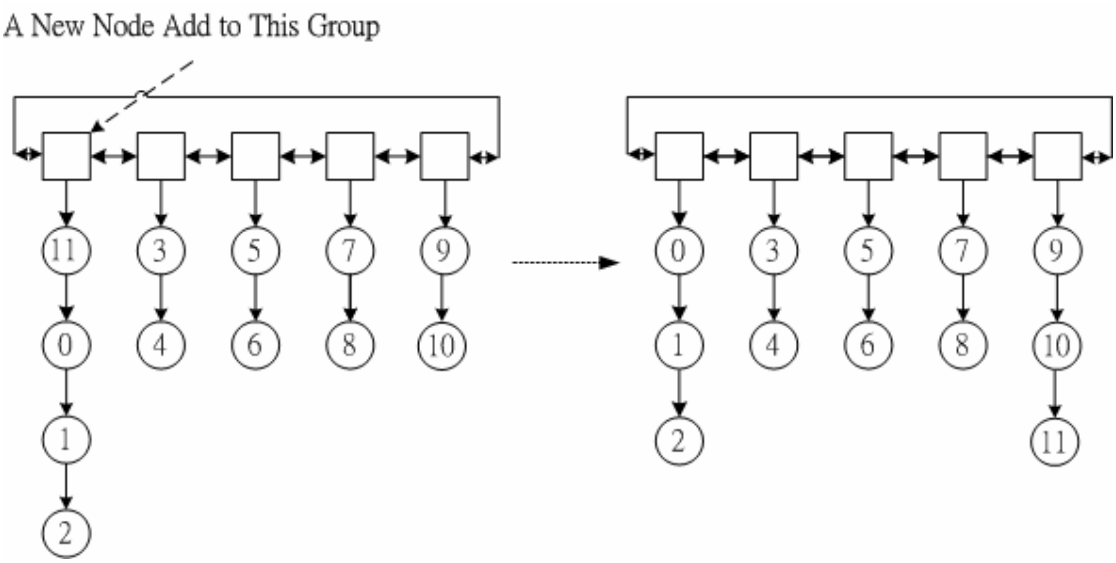


圖 3-19 Balance 示例

在 Add Balance 模式中，群組的 Common Node 移動數目取決於群組超過 Upper Bound 的數目和移動目標群組少於 Lower Bound 的數目，若是兩數相較前者較大或相等，則移動數目設定爲前者；反之移動數目設定爲群組超過 Lower Bound 的數目。Remove 模式中，群組的 Common Node 移動數目取決於群組少於 Lower Bound 的數目和調動目標群組多於 Upper Bound 的數目，若是兩數相較前者較大或相等，則移動數目設定爲前者；反之移動數目設定爲群組少於 Upper Bound 的數目。演算法如下：

Add Balance Algorithm:

```
if(no. of group > upper bound ){
        if(no. of group – upper bound >= lower bound – no. of target group){
                move no. = no. of group – upper bound;
        }
        else{
                move no. = no. of group – lower bound;
        }
}
```

```
else if(no. of group == upper bound && upper bound > lower bound &&
        no. of target group < lower bound){

    move no. = 1;
}
```

Remove Balance Algorithm:

```
if(no. of group < lower bound){
    if( lower bound – no. of group >= no. of target group – upper bound){
        move no. = lower bound – no. of group;
    }
    else
    {
        move no. = upper bound – no. of group;
    }
}
else if(no. of group == lower bound && upper bound > lower bound
    && no. of target group > upper bound){

    move no. = 1;
}
```

其中 no. of X 代表 Common Node Number of X，move no.代表所需移動的 Common Node 數目，move no.預設爲 0，若經過以上演算法運算後，move no.依然爲 0，則判定不需要進行 Balance，程序結束，否則依照 move no.數目將 Common Node 移往目標群組或是從目標群組調動過來，繼續往目標群組進行 Balance。

以上演算法的基本精神是將進行 Balance 的群組維持 Upper Bound 或 Lower Bound 的 Common Node 數量，且以移動數目較少爲原則，並且繼續往目標群組進行 Balance，直到不需進行 Balance 的群組爲止。經過 Balance 運作後，群組的 Common Node 數目分佈能大致成爲一個均衡的狀態。

Static Subscribe Server 管理的節點組織依照 Node Table 運算的結果分配，傳送 Command Message 給各群組的 Dynamic Subscribe Server，使其依照分配結果和 Common Node 聯繫，交換訊息。

圖 3-20 中，一開始僅有兩位使用者加入系統，接著第三位使用者加入，系統預設的初始 Max Group Number 爲 2，所以 3 個使用者超過，從其中選取 Efficiency 值較高者成爲新群組的 Dynamic Subscribe Server。接著第四位使用者加入，再度超過 Max Group Number，選取 Dynamic Subscribe Server 成立新的群組。此時群組數目和 Max Group Number 相等，Max Group Number 成長爲兩倍，Max Group Number = 2 x Max Group Number，變成 4。

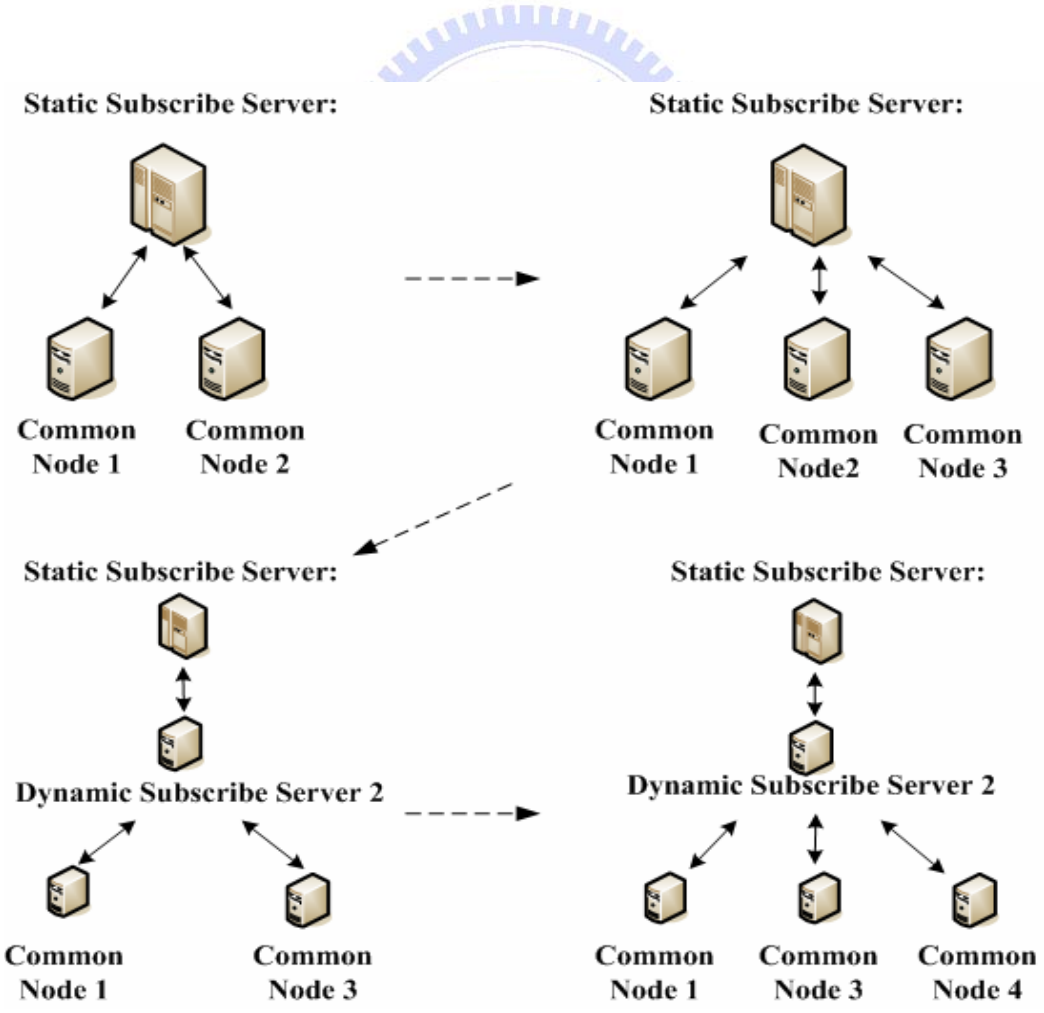


圖 3-20 節點管理範例 1

圖 3-21 中，第五位使用者 Common Node 6 加入，其後的數字假想爲 IP Address，第六位使用者 Common Node 5 加入後，會分配到 Dynamic Subscribe Server 4 的群組中，此時並未超過 Max Group Number，而在進行 Add Balance 的操作後，將 Common Node 3 搬移到 Dynamic Subscribe Server 2 的群組中。

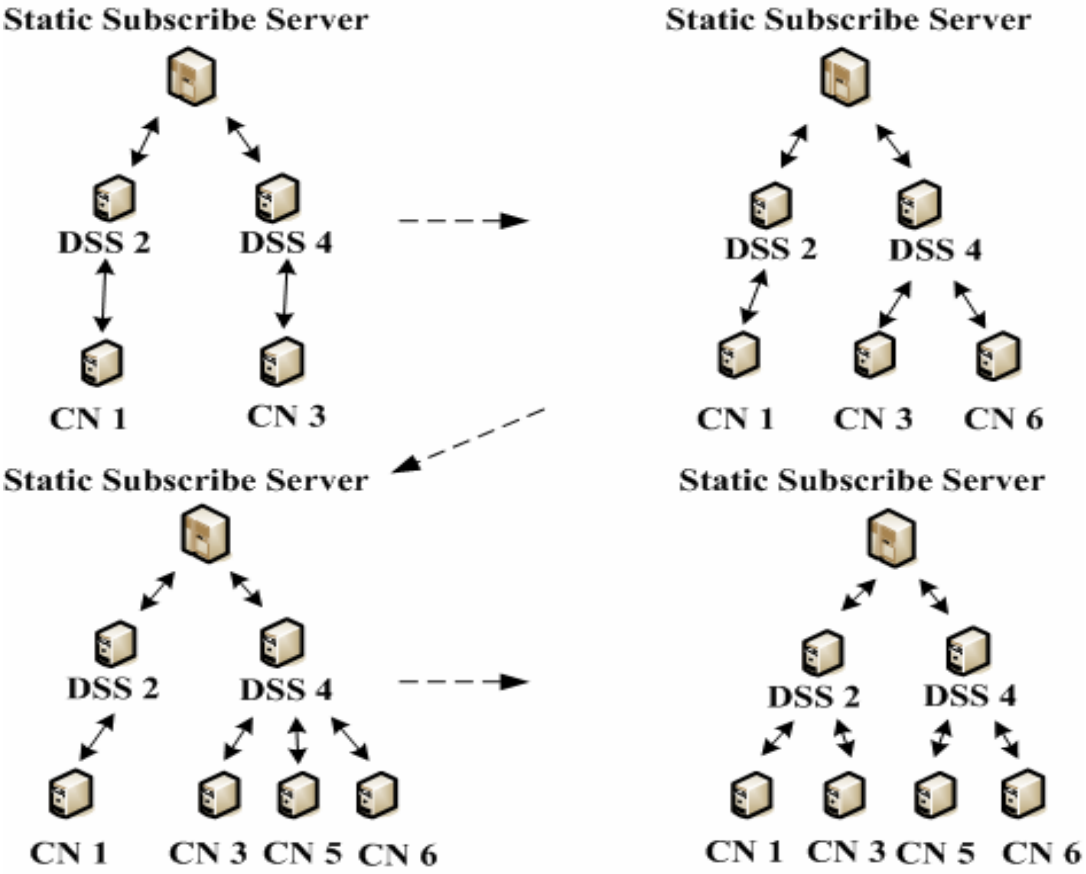


圖 3-21 節點管理範例 2

3.6.3 多重 Static Subscribe Server

我們的架構能讓使用者數量成長時，藉由群組和平衡負載的運作，系統仍能維持一定的效率，但是當使用者繼續增加時，爲了讓系統的負擔舒緩，以及服務更多的使用者，所以我們也設計了 Multi-Static Subscribe Server 的運作模式。

新 Static Subscribe Server 加入系統時，會和每一台舊有的 Static Subscribe Server 進行註冊，交換訊息，均衡分攤彼此所管理的群組。每台 Static Subscribe Server 擁有其他 Server 的相關資訊，存在 Server List 列表中，包含：IP Address, Message Port, Dno, MinIP, MaxIP。Dno 爲 Dynamic Subscribe Server 的數目，也代表此 Static Subscribe Server 所管理的群組數目，MinIP 表示其管轄的 IP Address 最小值，MaxIP 爲其所管轄的 IP Address 最大值，新使用者註冊時，就是依此範圍來判斷應該屬於哪個 Static Subscribe Server 管理。

新的 Static Subscribe Server 註冊時，由 Server List 列表中計算全部的群組數目，除以包含新 Server 在內的 Server 數，求得每台 Server 管理的群組平均數：

Avg. Group No. of each Server = All Group No. / No. of Servers

不能整除，則計算得到的餘數就代表前幾個 Server 需管理多一個群組，例如 9 個群組由 4 個 Server 分擔時，9 / 4 = 2 …… 1，則第一個 Static Subscribe Server 管理 3 個群組，其餘管理 2 個群組。

如圖 3-22 中，原來的 3 個 Static Subscribe Server 分別管理 G1~G3，G4~G6，G7~G9 群組，新 Static Subscribe Server 加入註冊，分別和舊有 3 個 Server 聯繫。第一個 Server 依然管理 G1~G3，不需變動。第二個 Server 管理 G4 和 G5，將 G6 交由第三個 Server 管理，第三個 Server 管理 G6 和 G7，將 G8 和 G9 交由新的 Static Subscribe Server 管理。

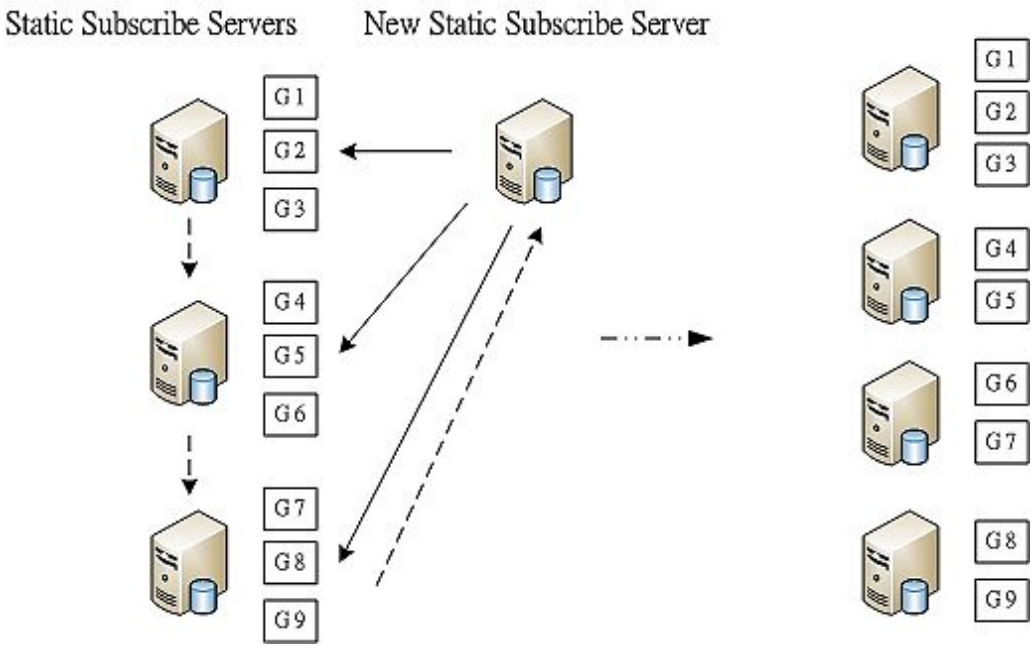


圖 3-22 新 Static Subscribe Server 註冊

新 Server 註冊完成後，即加入系統中服務。每個 Server 負責管理 MinIP~MaxIP 範圍內的節點，有新的 Group 產生時，會將新的 Dynamic Subscribe Server 資訊通知其他 Server。

3.6.4 金鑰管理

爲了安全性考量，在網路上傳遞的資訊使用密碼系統保密是非常重要的。我們的系統同時採用公鑰密碼系統和密鑰密碼系統，公鑰密碼系統安全性較高，但加解密速度慢，密鑰密碼系統速度較快，安全性稍差。

公鑰密碼系統必須產生一組 Key Pair，使用者在系統初始註冊時，產生 Key Pair，保存在本機端，並定時更新產生新的 Key Pair，以增加安全性。不將 Public Key 傳到系統的 Static Subscribe Server 端儲存，因爲每把 Key 所佔用的大小非常大，Server 端儲存的節點數量增加時，儲存 Public Key 所佔用的成本會相當大，

於是我們採用本機端自行保管的方式，當有需要使用到時，再彼此交換。

節點間的訊息傳遞，分爲三種型態：明文傳遞，公鑰密碼加密訊息，以及密鑰密碼加密訊息。當傳遞訊息時，會先傳送明文的 Message Type，表明此次傳輸所使用的型態，0 代表密鑰密碼加密訊息，1 代表使用公鑰密碼加密訊息，2 代表明文訊息，由此確認收到的訊息型態，再使用相對應的密碼解密或是直接讀取。傳送的訊息需要使用密碼加密時，先以明文型態的 Command Message 和對方要求 Public Key，取得之後，再利用此 Public Key 交換 Secret Key，或是直接使用 Public Key 進行加密。

Static Subscribe Server 和 Slave Dynamic Subscribe Server 之間，以及 Dynamic Subscribe Server 和 Slave Common Node 之間，會預先利用 Public Key 進行 Secret Key 的交換，之後訊息間的傳遞就使用 Secret Key 進行加密保護，且週期地由下層的節點更新交換 Secret Key。

其餘節點之間，由彼此詢問交換取得 Public Key，加密傳遞訊息。因爲變動性大，每個節點可能會面對的對象非常多，彼此交換 Secret Key 會造成儲存 Secret Key 的負擔過大，且節點之間可能過很久才會再度傳遞訊息，則儲存彼此之間的 Secret Key 變成浪費儲存空間的工作。使用 Public Key 加密節點之間溝通的訊息，由於訊息的長度不會太大，加密所花費的時間也不會造成太大的影響。

進行匿名傳輸時，由於傳遞的資料量可能非常龐大，因此必須使用 Secret Key 進行加密。傳輸路徑節點之間使用的 Secret Key 會在 Tunnel Setup 時分配好，在下一章中描述。

3.6.5 通道建立

在我們的系統中，Tunnel Setup 選取的節點由 Client 端自行選取，可以確保較高的隱私性。有些系統的設計是 Client 端送出匿名傳輸的請求給系統後，直接由系統的 Server 幫 Client 決定好路徑，如此 Server 端就能夠知道 Client 端的通訊對象，若是其中有攻擊者潛入或是有被買通的共謀份子，或系統方本身就是不懷好意，則 Client 的通訊隱私就很容易暴露。

Client 會先和 Dynamic Subscribe Server 取得匿名網路中節點的資訊，並將這些資訊存在 Node Table 之中，其中除了所取得的節點資訊之外，也會依照取得的 Dynamic Subscribe Server 來源分開存放。進行 Tunnel Setup 時，從不同的 Dynamic Subscribe Server 來源各選取一個節點，選出 Client 設定或是系統預設的匿名路徑節點數目個節點，例如 Client 端的匿名路徑節點數設定是 3，就會從 Node Table 之中不同的來源間選出 3 個來源，並各自從這 3 個來源中選出 1 個節點，進行 Tunnel Setup，如圖 3-23。

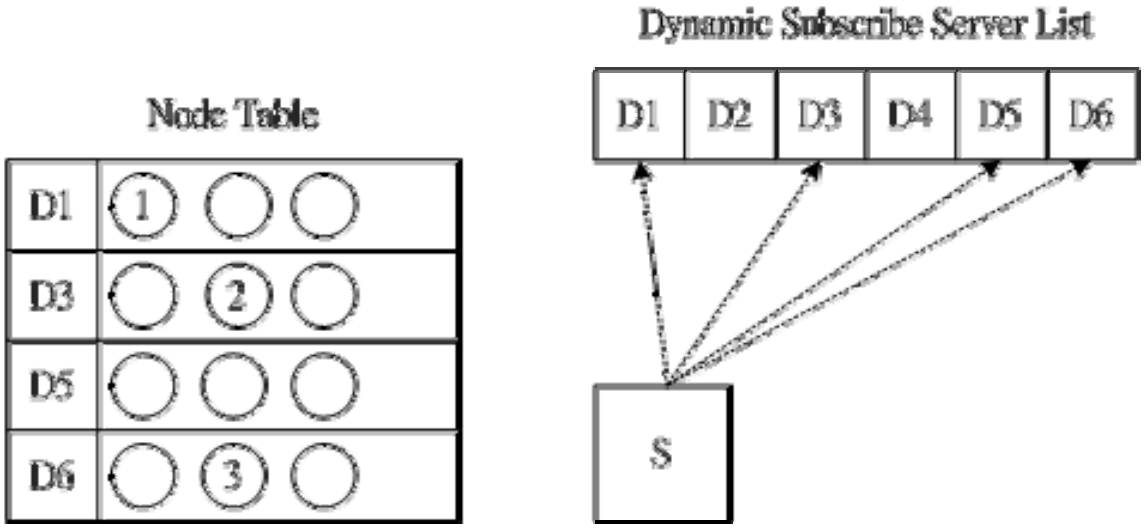


圖 3-23 Tunnel Setup 節點選取

選取完節點之後，由 Client 端主動去向各個節點取得 Public Key，並且對各節點產生一把 Secret Key 以及一個 Serial Number。封裝 Tunnel Setup 封包如圖 3-24。

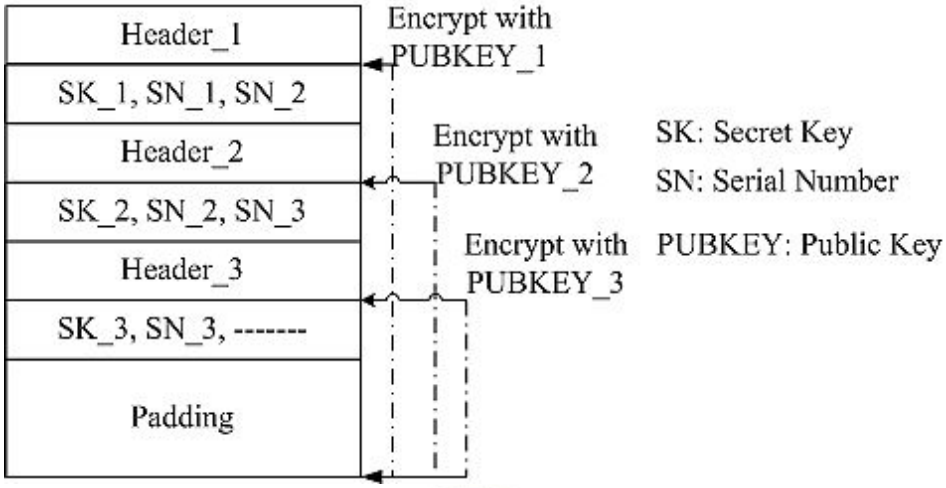


圖 3-24 Tunnel Setup 封包

假設 Client 選定節點 1，2，3 爲 Tunnel 中的節點，則將 Tunnel Setup 封包送往節點 1。節點 1 接收後，用 Private Key 解開，記錄下 SN_1, SK_1, SN_2，如圖 3-25。將 Packet 繼續往節點 2 傳送，節點 2 收到後用 Private Key 解開，記錄下 SN_2, SK_2, SN_3，如圖 3-26。將 Packet 往節點 3 傳送，節點 3 收到後用 Private Key 解開，記錄下 SN_3, SK_3，如圖 3-27。

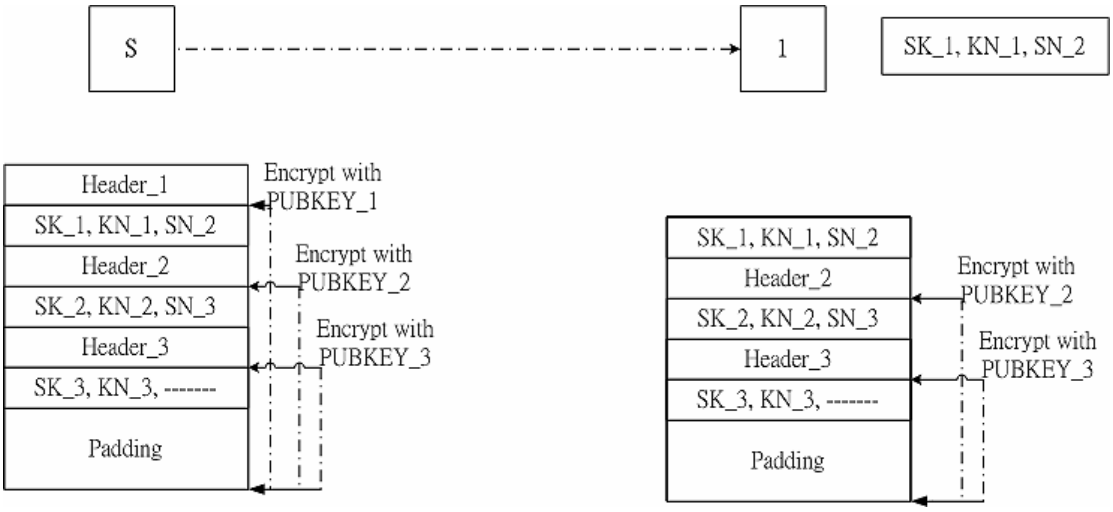


圖 3-25 Tunnel Setup 示例 1

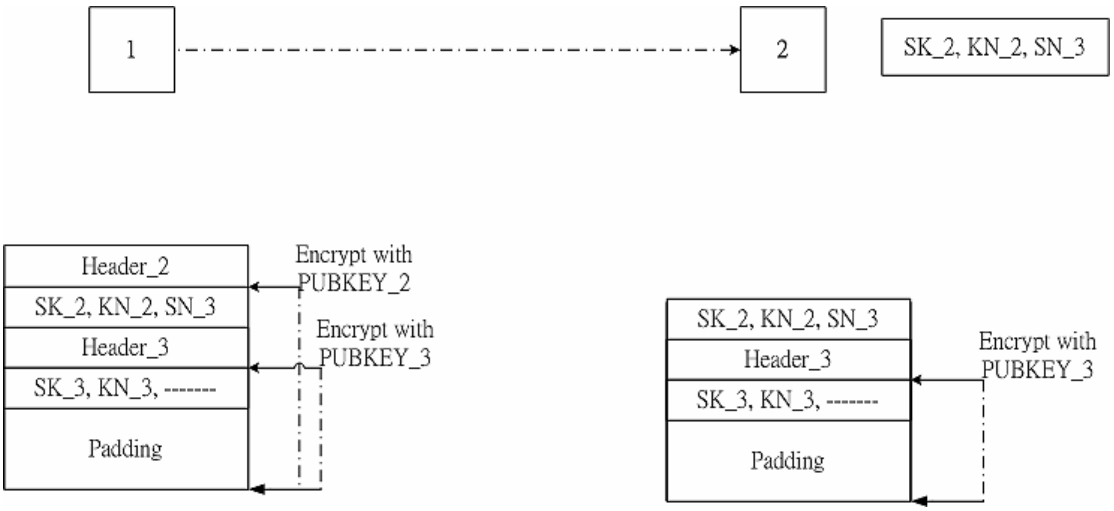


圖 3-26 Tunnel Setup 示例 2

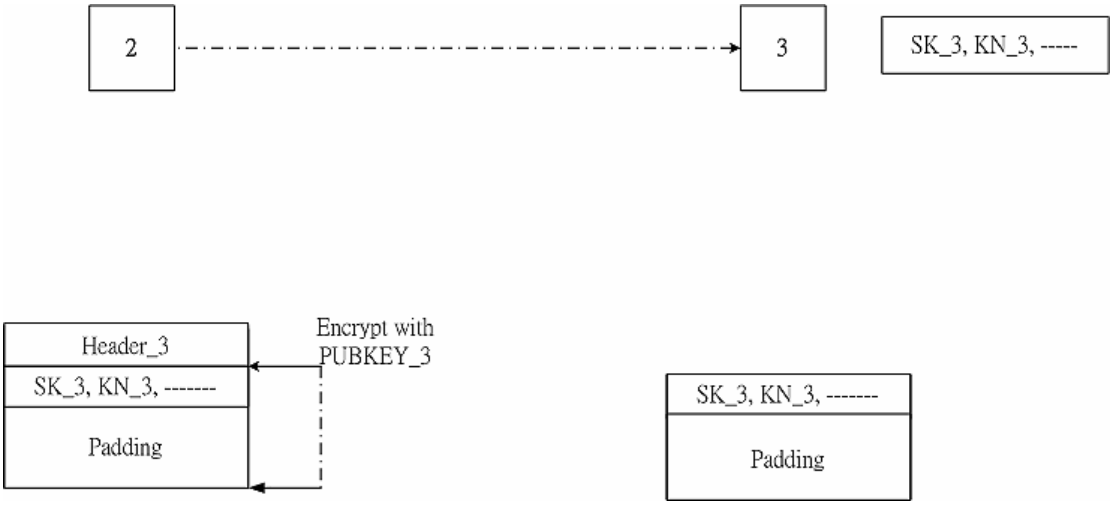


圖 3-27 Tunnel Setup 示例 3

其中SK_1 爲Client之後進行匿名傳輸時，節點 1 轉傳Onion Packet時解密所用的Secret Key，而SN_1 爲用來比對出要使用SK_1 解密的一個指示記號，將解密後的Onion Packet的SN欄位填寫上SN_2 後，繼續往節點 2 傳送[4]。如此在Tunnel Setup的過程中，每一個節點都不知道此Tunnel的建立者，可以確保傳送端的隱私。

3.7 流程

以下爲使用者啓動 Client 端程式到進行匿名傳輸的完整流程：

1. 調整設定 Client 端程式參數。
2. 啓動 Client 端應用程式，進行初始化後向 Static Subscribe Server 進行註冊。
3. Client 端向 Master Dynamic Subscribe Server 請求 Dynamic Subscribe Server List。
4. Client 端向 Dynamic Subscribe Server List 中的 Dynamic Subscribe Server 請求節點資訊，存於 Node Table 中。
5. 從 Node Table 中選取節點，進行 Tunnel Setup。
6. 經由 SOCKS Protocol，將本機端網路應用程式傳輸的封包導入本機端運行的 Anonymous Socks Server。
7. 經過層層封包加密處理後，產生 Onion Packet，向 Tunnel 中的下一站傳送。
8. 中間的轉傳節點收到 Onion Packet，將其解開，處理後往下一站傳送。
9. 最後一層 Onion Packet 解開後，接收者得到真正的傳輸封包，完成匿名傳輸。

第四章 成果與分析

4.1 環境設置

本篇論文主要是設計一個匿名網路系統的節點架構，希望能在系統成本負擔不大的情形下，依然能夠有效率地服務大量使用者，使得本系統的匿名傳輸功能得到充分發揮。

我們的系統開發平台爲 Linux / FreeBSD 等 UNIX 作業系統，爲了實際測試實驗成果，除了本實驗室和校內的電腦之外，也向全台各地許多學校商借電腦主機和網路資源，做爲系統中實際參與運作的網路節點，列於表 4-1，在此再次鄭重感謝！

Location	**IP Address**	**Operating System**	**Efficiency**
交大電資	140.113.179.205	Ubuntu Linux	CPU: P3-500 Mem: 128MB
交大電資	140.113.179.206	Ubuntu Linux	CPU: P3-500 Mem: 128MB
交大電資	140.113.179.207	Ubuntu Linux	CPU: P3-500 Mem: 128MB
交大工三	140.113.166.24	FreeBSD 4.11 Stable	CPU: P4-1.6G Mem: 768MB
交大工三	140.113.166.25	Ubuntu Linux	CPU: AMD 1800+ Mem: 768MB
交大工三	140.113.166.31	FreeBSD 4.10 Stable	CPU: AMD 900 Mem: 256MB
清大資工	140.114.79.112	Ubuntu Linux	CPU: AMD 3000+ Mem: 512MB

中央資工	140.115.50.36	FreeBSD 6.2 Stable	CPU: P3-1G Mem: 256MB
中央資工	140.115.50.53	FreeBSD 6.2 Stable	CPU: P3-735 Mem: 256MB
中山電機	140.117.164.33	RedHat Linux	CPU: AMD 900 Mem: 256MB
中山電機	140.117.167.236	RedHat Linux	CPU: P3-667 Mem: 192MB
政大資科	140.119.164.48	FreeBSD 6.2 Stable	CPU: P3-1.0 G Mem: 512MB
中興電機	140.120.108.93	RedHat Linux	CPU: AMD 3500+ Mem: 3GB
中華資工	140.126.130.33	Fedora Linux	CPU: AMD 2800+ Mem: 128MB
中華資工	140.126.130.34	Fedora Linux	CPU: AMD 2800+ Mem: 128MB
中華資工	140.126.11.2	Fedora Linux	CPU: AMD 2800+ Mem: 443MB
東華資工	134.208.3.103	Ubuntu Linux	CPU: P4 - 1.8G Mem: 256MB
東華資工	134.208.3.107	Ubuntu Linux	CPU: Celeron - 1.2G Mem: 1036MB

表 4 -1 實驗資源

4.2 實驗方法

本篇論文以系統架構設計爲主，節點的組織管理和訊息交換等等，負責進行匿名網路傳輸的前置工作，主要的實驗目標爲測試此架構的運作情形是否如預期中進行，包括節點的分佈狀態，以及其中的運行機制的檢視。主要的實驗進行以資源列表中的主機爲主，部份測試爲了檢驗大量節點的運作狀況，使用模擬的網路節點進行測試。

有關本系統匿名傳輸的效能，網路傳輸資料的匿名性檢驗，路徑追蹤機制實測效果，在"匿名網路之管理與連線分析"[4]這篇論文中會有詳細的實驗情形和數據分析。

4.3 成果展現

4.3.1 節點組織

爲了測試 Balance 機制，以及節點的分佈狀態，節點加入時 Node Table 的處理效率等情形，我們先採用模擬的方式加入節點，觀察 Node Table 的運作：

模擬資源列表中的節點：

```
no. of nodes : 18  no. of groups : 4  |  max no. of each group: 8
------------------------------------------------------------------
list 0: 140.115.50.36 13288 372323304 0 232 - 4 nodes
------------------------------------------------------------------
  1: 140.126.130.33 18501 1423067205 69
  2: 140.126.130.34 17257 1623409513 105
  3: 134.208.3.103 18252 2072135500 76
  4: 134.208.3.107 28319 197619359 159

list 1: 140.113.166.24 37534 1596428958 0 158 - 4 nodes
------------------------------------------------------------------
  1: 140.113.166.25 39116 2055903436 204
  2: 140.113.166.31 35272 1935903176 200
  3: 140.113.179.205 63293 53212989 61
  4: 140.113.179.206 60905 1769336297 233

list 2: 140.113.179.207 37363 1891537395 0 243 - 3 nodes
------------------------------------------------------------------
  1: 140.114.79.112 24049 675438065 241
  2: 140.115.50.53 9255 397878311 39
  3: 140.117.167.236 38682 1160484634 26

list 3: 140.117.164.33 4002 1137512354 0 162 - 3 nodes
------------------------------------------------------------------
  1: 140.119.164.48 58423 840950839 55
  2: 140.120.108.93 40509 526491197 61
  3: 140.126.11.2 26915 1302751523 35
```

圖 4-1 Node Table 模擬 1

圖 4-1 爲模擬資源列表節點的結果，其中最上方依序爲所有節點的數目，群組數目，每個群組的最多 Common Node 節點數。List 即代表一個群組，List 同列爲 Dynamic Subscribe Server，之下爲群組內的 Common Node。編號除外，依序爲 IP Address，Message Port， Authentication Number，以及 Efficiency。由表中觀察呈現均衡分佈的狀態，位於交通大學的網路節點佈於同一群組中。

隨機產生 1000 個節點：

約進行五十次，每次均產生 31 個群組，每個群組 Common Node 節點數目都在 30~32 個節點之間，呈均衡分佈，圖 4-2 爲其中某一群組的節點資訊截圖。

```
list 14 : 50.151.123.229 31717 848788453 0 229 - 32 nodes
-----------------------------------------------------------------
  1: 57.104.185.104 47464 963164520 104
  2: 57.110.151.129 38785 963549057 129
  3: 57.154.144.168 37032 966430888 168
  4: 57.156.50.109 12909 966537837 109
  5: 57.194.127.237 32749 969048045 237
  6: 57.229.1.135 391 971309447 135
  7: 58.7.215.80 55120 973592400 80
  8: 58.104.229.139 58763 979953035 139
  9: 58.108.172.194 44226 980200642 194
 10: 58.185.68.217 17625 985220313 217
 11: 58.224.191.136 49032 987807624 136
 12: 59.21.105.147 27027 991259027 147
 13: 59.76.180.83 46163 994882643 83
 14: 59.86.151.157 38813 995530653 157
 15: 59.231.135.136 34696 1005029256 136
 16: 59.253.22.194 5826 1006442178 194
 17: 60.10.137.62 35134 1007323454 62
 18: 60.18.229.174 58798 1007871406 174
 19: 60.41.119.173 30637 1009350573 173
 20: 60.59.56.187 14523 1010514107 187
 21: 60.60.153.163 39331 1010604451 163
 22: 60.96.98.210 25298 1012949714 210
 23: 60.101.206.91 52827 1013304923 91
 24: 60.139.63.114 16242 1015758706 114
 25: 60.141.232.246 59638 1015933174 246
 26: 60.181.247.221 63453 1018558429 221
 27: 60.199.254.93 65117 1019739741 93
 28: 61.24.111.253 28669 1025011709 253
 29: 61.24.132.93 33885 1025016925 93
 30: 61.156.184.116 47220 1033681012 116
 31: 61.171.18.57 4665 1034621497 57
 32: 61.193.143.135 36743 1036095367 135
```

圖 4-2 Node Table 模擬 2

隨機產生 10000 個節點：

約進行五十次，每次均產生 78 個群組，每個群組 Common Node 節點數目都在 126~128 個節點之間，呈均衡分佈，圖 4-3 爲部份節點資訊截圖：

```
list 73 : 93.180.169.255 43519 1572121087 0 255 - 127 nodes
--------------------------------------------------------------------------------
  1: 119.115.123.248 31736 2004057080 248
  2: 119.115.213.225 54753 2004080097 225
  3: 119.118.234.56 59960 2004281912 56
  4: 119.123.184.210 47314 2004596946 210
  5: 119.124.162.169 41641 2004656809 169
  6: 119.124.203.91 52059 2004667227 91
  7: 119.127.141.104 36200 2004847976 104
  8: 119.128.184.232 47336 2004924648 232
  9: 119.132.105.110 26990 2005166446 110
 10: 119.133.59.73 15177 2005220169 73
 11: 119.140.156.27 39963 2005703707 27
 12: 119.142.131.109 33645 2005828461 109
 13: 119.143.24.35 6179 2005866531 35
 14: 119.144.28.178 7346 2005933234 178
 15: 119.145.233.148 59796 2006051220 148
 16: 119.150.205.197 52677 2006371781 197
 17: 119.153.250.115 64115 2006579827 115
```

圖 4-3 Node Table 模擬 3

真實加入網路節點：

圖 4-4，啓動 Static Subscribe Server，等待節點加入。

圖 4-4 Static Subscribe Server 啓動

圖 4-5，Client 啓動程式執行，成功加入系統。

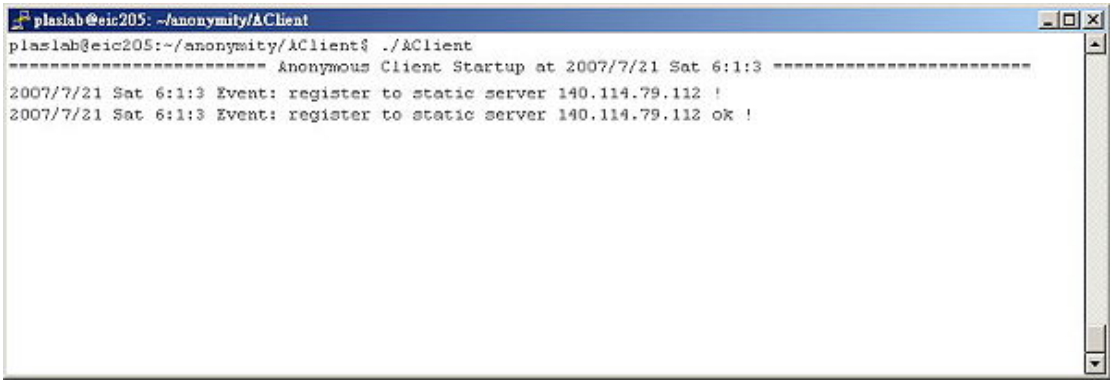


圖 4-5 使用者啓動 Client 程式，進行註冊

圖 4-6，第三位使用者加入，超過預設的群組最大值，從原來的三個節點中選取 Efficiency 值最高者，因爲這三台電腦的設備都一樣，算出來的值相同，選取 IP 值最小者成爲新的 Dynamic Subscribe Server。

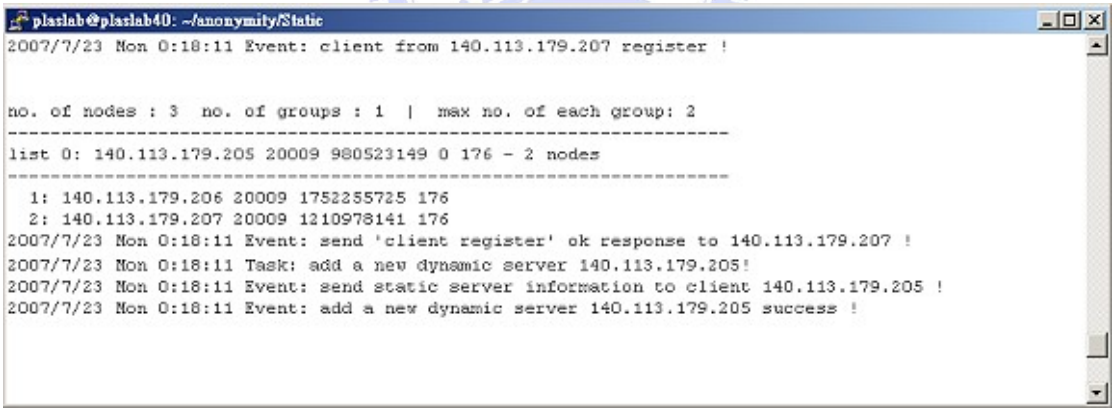


圖 4-6 建立新群組 1

圖 4-7，另兩位使用者繼續加入，再度形成新的群組，此時因爲群組數目到達原來的群組最大值，故群組最大值倍增爲兩倍由 2 變爲 4。

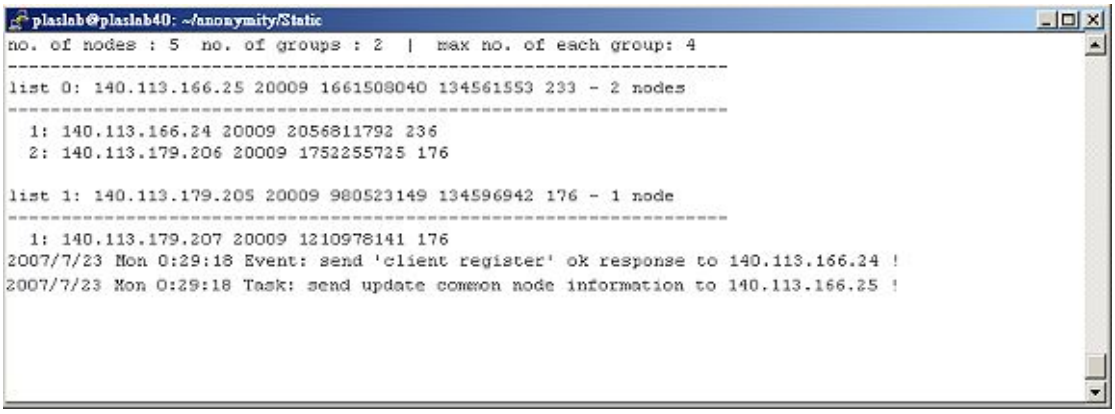


圖 4-7 建立新群組 2

圖 4-8、圖 4-9 爲 Dynamic Subscribe Server 端的畫面。

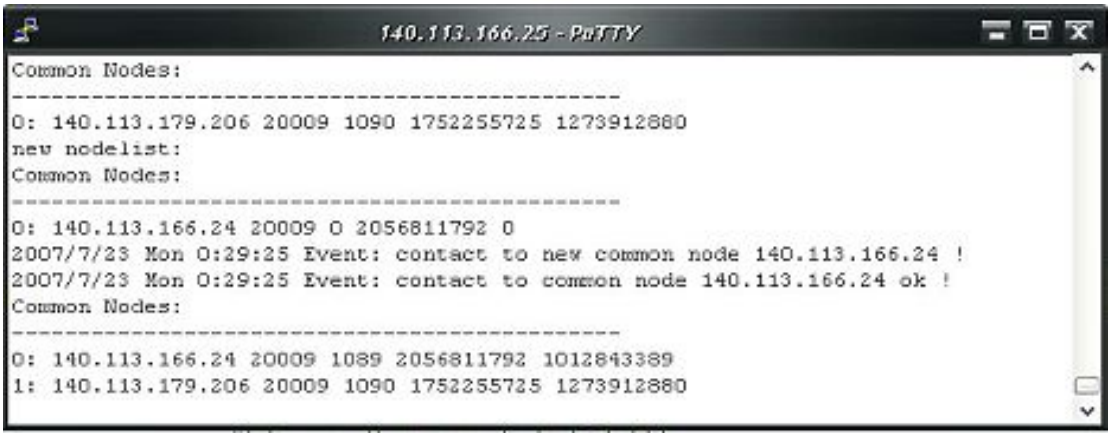


圖 4-8 Dynamic Subscribe Server 1

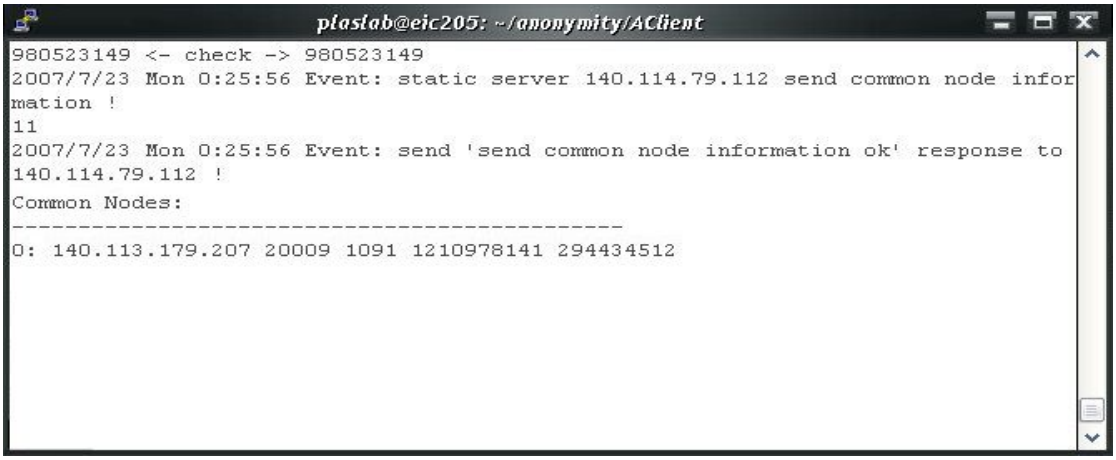


圖 4-9 Dynamic Subscribe Server 2

圖 4-10 爲逐步加入節點之後，最後形成類似圖 4–1 的模擬結果，因部份主機實驗時無法正常運作，故實測畫面相較之前模擬節點數量少。

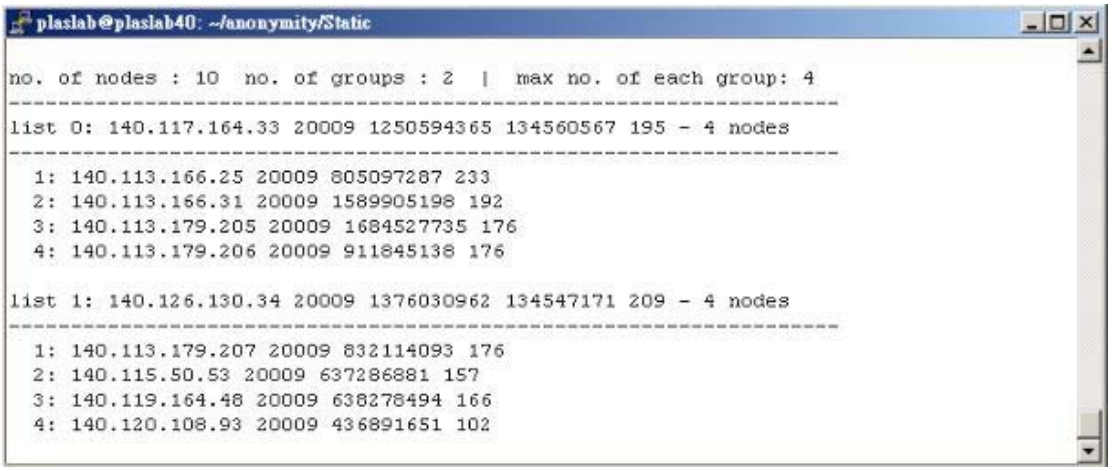


圖 4-10 節點加入最終情形

4.3.2 容錯測試

使用者除了正常的運作之外，也有可能出現非正常管道離開系統的情形，必須偵測出這些現象，將不存在的節點資訊清除，使其不影響系統運作。Client 本身會向上層回報自己的運作狀態，若超過一定時間沒有回報，上層將此節點清除，並回報給系統端。

由 Static Subscribe Server 直接處理的節點移除：

圖 4-11，Client 端終止程式運行。

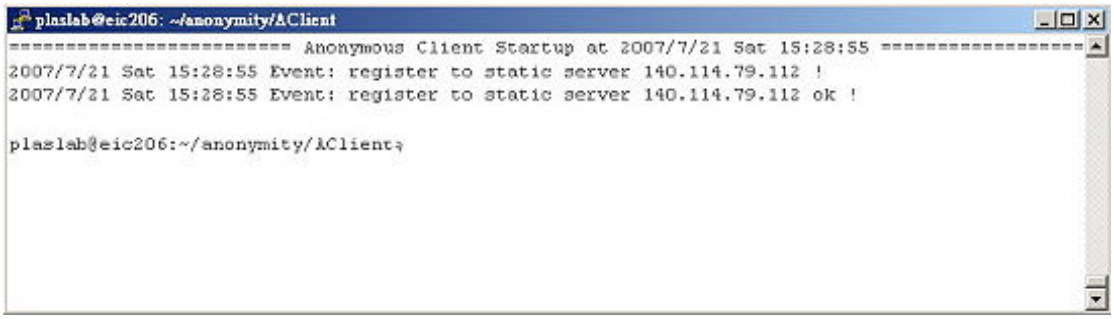


圖 4-11 Client 離開系統

圖 4-12，Server 端偵測到使用者超過期限未回報，主動探詢後，將其移除，圖中 Static Subscribe Server 原有兩個 Common Node 的資訊，偵測後發現編號 2 號的 Client 已經離開系統，將其移除。

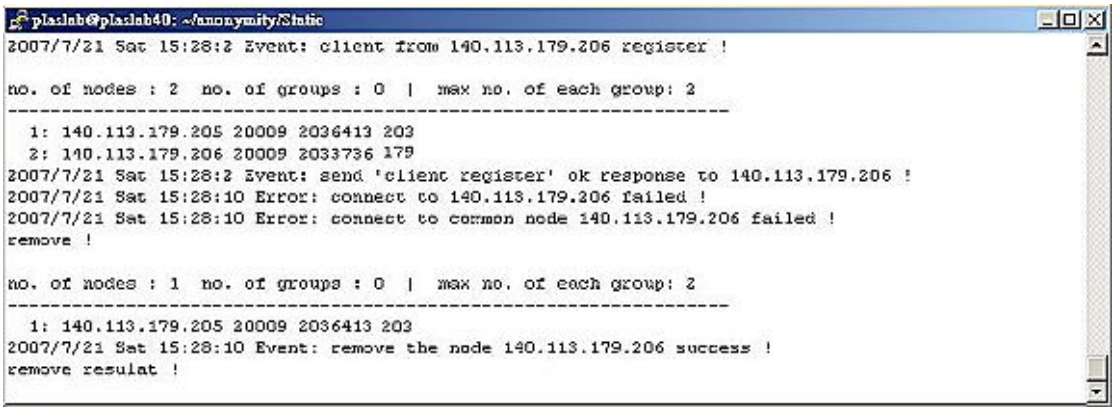


圖 4-12 Static Subscribe Server 移除不存在的節點

由 Dynamic Subscribe Server 回報的節點移除：

圖 4-13 爲目前的節點狀態，之後節點 140.113.179.206 離開系統。

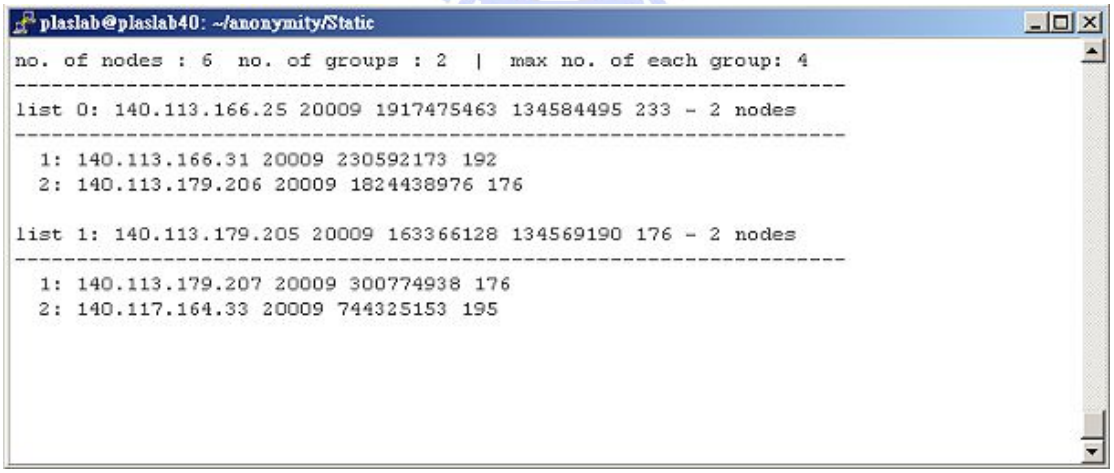


圖 4-13 Dynamic Subscribe Server 回報節點移除 1

圖 4-14，超過回報週期的兩倍時間，Dynamic Subscribe Server 140.113.166.25 回報 Static Subscribe Server，進行節點移除。

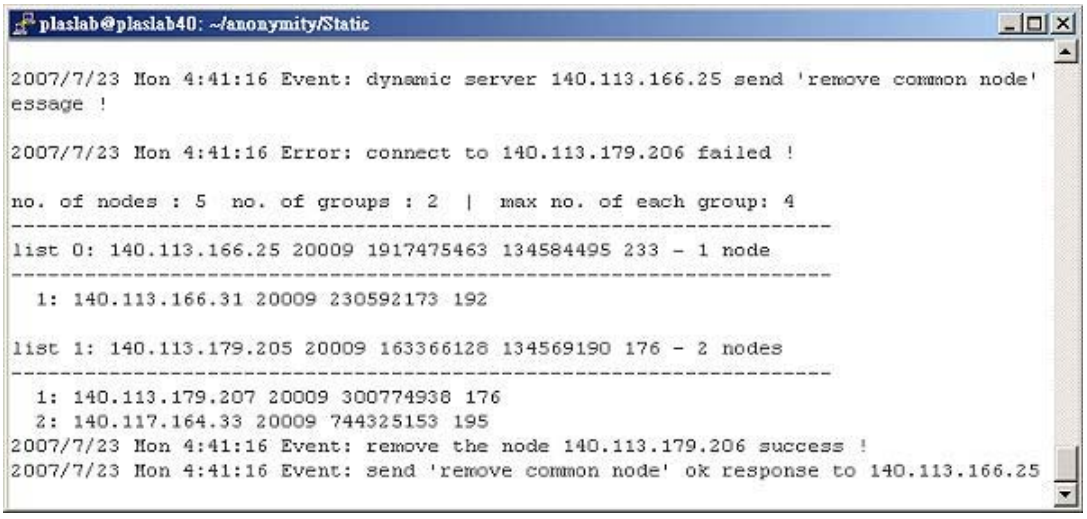


圖 4-14 Dynamic Subscribe Server 回報節點移除 2

移除 Dynamic Subscribe Server：

圖 4-15 為目前系統的節點狀態，之後 Dynamic Subscribe Server 140.113.166.25 離開系統。

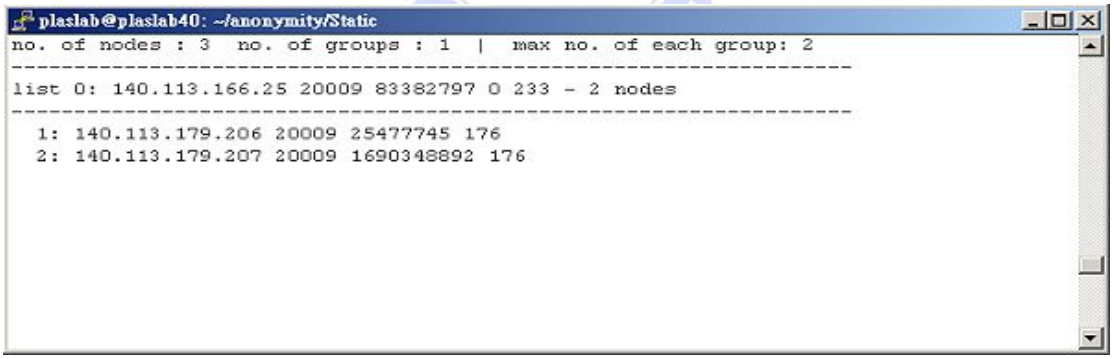


圖 4-15 移除 Dynamic Subscribe Server 1

圖 4-16，Static Subscribe Server 偵測到 Dynamic Subscribe Server 已經不在系統中運行，將其移除。

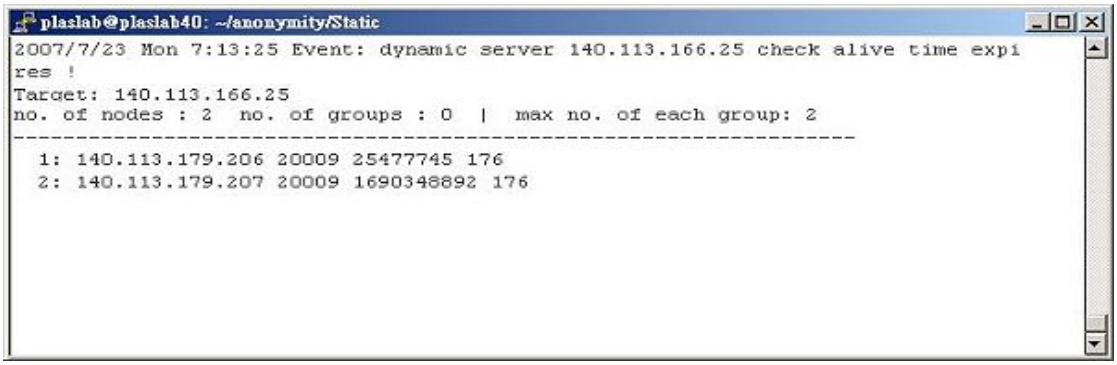


圖 4-16 移除 Dynamic Subscribe Server 2

4.4 系統比較

我們的系統架構採用 Fixed Server 和 Peer-to-Peer 融合的方式，針對 Low-Latency 的網路應用提供匿名通訊的服務。目前的匿名網路系統中，同樣也可提供 Low-Latency 匿名服務的有 Anonymizer、Tor、Tarzan……等，我們在此對其簡介並與之比較。

Anonymizer

Anonymizer 是一個提供匿名網路服務的商業網站，曾經與美國政府的美國之音合作，目的爲幫助中國人民使用匿名通訊上網，避免中國政府的監督，以保障網路的隱私和言論自由，目前仍然與美國之音合作，幫助伊朗民眾使用匿名通訊免於政府的網路監控。

Anonymizer 使用 Proxy Server 的方式提供使用者匿名瀏覽網頁，使用者登入 Anonymizer 網站，輸入目標網址，由它取得網頁內容再回傳給使用者。因此目標網站管理者僅能得知 Anonymizer 所提供的 Proxy Server 位置，無法得知使用者的真確位置，網路監控攻擊者也只能夠觀察到使用者和 Anonymizer 公司的連線，無法得知真正的目標網頁。在網頁資料的傳輸中，爲了保障安全性，使用 SSL(Secure Socket Layer）協定進行加密。

Anonymizer 主要是針對 HTTP 協定的網頁瀏覽提供匿名性的保障，我們的系統則可在所有使用 TCP 協定的網路應用中使用；Anonymizer 使用傳統中央式 Server 的架構，對於使用者數量快速成長時，在系統端的負擔必然也會隨之快速增加，我們系統採用半 Peer-to-Peer 的架構，使用者可以同時分攤整個系統的運作成本；Anonymizer 採用 Single Proxy 的方式，匿名傳輸的路徑僅通過單一 Proxy 轉傳便通往目的端，攻擊者僅需要觀察此 Proxy 進出的網路流量，對於分析真實路徑的複雜度較低，分析出的機會較高，而且只要滲透或買通系統方的人員，則此次匿名傳輸路徑就輕易得知了，我們系統採用多點繞徑的方式，預設透過三個中間轉傳節點，增加攻擊者分析的困難度，而且其中的轉傳節點也無法得知整體匿名傳輸的路徑情形，可避免匿名路徑輕易就被系統中的惡意節點洩漏出；此外，Anonymizer 是商業公司，要使用匿名傳輸必須付費，而我們系統是以免費公開的形式運作，任何人只要有需要進行匿名傳輸，都可透過我們的系統，不需要付費】。

Tor

Tor全名爲The Onion Router，是第二代Onion Routing系統，使用者經由Tor設置，可以在網路上進行匿名通訊。Tor最初由美國海軍研究實驗室(US Naval Research Laboratory)贊助，之後成爲EFF(Electronic Frontier Foundation)的一個研究項目，目前EFF不再贊助，Tor依然持續運作和研究開發。[19]

使用者安裝 Tor 提供的 Client 端應用軟體，和系統的 Directory Server 聯繫取得 Onion Server 的資訊，便可透過任意 3 個 Onion Server 形成 Circuit，進行匿名網路通訊，使用者也可運行 Server 軟體，成爲 Onion Server。Tor 在 Application Layer 進行傳輸之間的加密，Onion Server 之間使用 TLS(Transport Layer Security)進行加密，針對 TCP Stream 提供匿名傳輸的功能。

我們的系統和 Tor 相似處較多，同樣是針對 TCP 網路應用提供匿名傳輸，但在系統的架構上，Tor 主要還是傳統式的中央 Server-Based 架構，主要 Server 由系統端提供，而使用者可自願運行 Server 程式加入系統中服務，我們系統是採用半 Peer-to-Peer 架構，使用者進入系統後，會依據系統端運算後的網路狀態進行組織，當使用者數量增加成長時，由使用者扮演的 Server 也會隨之成長，而系統僅需要提供少量負責使用者節點註冊和組織網路狀態的 Server，且這些 Server 是隨機動態的，隨著週期時間和節點數量的成長變化，較不容易成爲攻擊者鎖定的目標；在轉傳節點的選取上，Tor 主要是由數個 Tor Server 負責轉傳封包，而我們的系統則是每位使用者節點都有可能擔負封包轉傳的角色，在轉傳節點的選擇上能提供較大的數量和較高的複雜度。

Tarzan

Tarzan 是一個完全 Peer-to-Peer 架構的匿名網路系統，不需要中央的 Server，每個使用者運行 Client 端程式後，自動尋找其他系統中的使用者，彼此交換訊息。每個使用者都同時具有 Client／Server 身份，共同分擔系統的運作。

當使用者節點進入系統後，使用Gossip-based的Protocol[20]，進行Peer Discovery，取得系統中所有其他節點的資訊。之後進行Tunnel Setup的運作，從本身所取得的節點資訊中選取數個節點，層層封裝Tunnel Setup封包後，依序傳遞到選取的節點進行資訊交換，建立Tunnel。

純Peer-to-Peer系統在進入系統後必須先進行Peer Discovery以取得其他的網路節點資訊，較爲耗時且浪費頻寬，且缺少中央控管的機制，我們的系統可以在使用者節點進入之後，很快向 Server 取得其他的節點資訊；在 Tarzan 的設計上，每個節點必須要取得所有其他網路節點的資訊，當系統規模漸趨龐大時，對於每

個節點來說，會造成儲存這些資訊的負擔，若是僅儲存部份節點資訊，則又造成匿名傳輸路徑複雜度降低的傷害，我們的系統將節點資訊分散在數個由使用者扮演的 Dynamic Subscribe Server 之中，每個節點動態地去和這些節點更新節點資訊，使用者可以自行設置調整儲存其他節點資訊的數量；此外，Tarzan 設計上要每個節點取得所有其他節點的資訊，因此攻擊者進入系統後，便可藉由 Peer Discovery 取得所有系統中節點的資訊，理論上攻擊者便可藉由觀察所有節點的網路流量進行分析，造成威脅，我們的系統是由 Dynamic Subscribe Server 負責回應節點資訊的請求，而每次的回應僅將自己所管理的節點資訊的部份傳送給請求節點，亦可再加上一些更有效的控管機制，避免攻擊者散佈的惡意節點在短時間內輕易取得系統中所有的節點資訊。

4.5 安全性分析

關於系統的安全性，我們分成以下各點來加以分析說明：

訊息加密

在系統中傳遞的各種控制訊息，利用現有加密技術來加以保護，避免攻擊者攔截竊取其中的內容，取得重要訊息，於3.6.4中有詳細的描述。

管理組織

爲了避免攻擊者發送假冒的訊息干擾或破壞系統的組織架構，在每個使用者進行註冊時，會產生 Authentication Number，傳送到 Static Subscribe Server，再傳送給使用者所屬的 Dynamic Subscribe Server。當 Dynamic Subscribe Server 發送指令訊息給使用者時，必須包含此 Authentication Number，使用者接收訊息後，比對相同，才依照指令行事。另外，當 Common Node 轉變成 Dynamic Subscribe Server 時，會產生 Request Number，並傳送給 Static Subscribe Server。使用者取

得 Dynamic Subscribe Server 列表時，其中也包含列表中各 Server 的 Request Number，當使用者發送請求節點訊息時，包含 Server 的 Request Number，當 Server 接收訊息後比對，與本身的 Request Number 相同後才接受此訊息，並回應節點資訊，避免攻擊者非法取得節點資訊。

節點選取

匿名路徑中的轉傳節點，皆由使用者自行選取，避免第三方得知此訊息，而洩漏出完整路徑。當使用者向 Dynamic Subscribe Server 取得節點資訊後，再自行從其中選出所要進行匿名傳輸的轉傳節點，所以即使是 Dynamic Subscribe Server，也無法明確得知各使用者所選取的路徑，而且使用者向多個 Dynamic Subscribe Server 要求部份節點資訊，再從其中部份 Server 所取得的節點中各挑選一個成爲轉傳節點，避免選取的轉傳節點都由同一個 Server 中選取，降低某一 Server 得知完整路徑的可能性。而且相近網域的節點會群聚於相近的群組中，從不同 Server 所掌控的群組中選取節點，可以增加匿名路徑跨越的網域廣度，增加攻擊者觀察分析的困難度。

通道建立

轉傳節點選取好之後，通道建立是由使用者節點封裝好 Tunnel Setup Packet，一站站傳遞下去，則在路徑中的每個節點只知道路徑中的前一站和後一站，無法得知完整路徑，也無法確知匿名傳輸的發起者，如此可避免攻擊者在系統中掌控的惡意節點，被選取爲匿名路徑中的轉傳節點時，輕易得知完整路徑的危險。通道建立描述於 3.6.5 中。

第五章 結論與未來展望

5.1 結論

由先前的介紹和實驗結果來看，我們所設計的系統架構可以讓節點組織成均衡的分佈狀態，使用者分攤系統的負擔，當系統規模增大時，依然能維持良好運作。每個節點都能有效率地取得系統中其他節點的資訊，也能避免惡意攻擊者短時間內取得所有節點的資訊，造成同時觀察分析所有節點而帶來的威脅。而將相近網域的節點限制在相近的群組中，使得從不同群組取得的節點資訊所建立的 Tunnel 所跨越的網域更廣，路徑更複雜，更不易分析。而攻擊者本身所散佈的惡意節點若是也屬於相近的網域中，也能將其叢聚在相近的群組減少感染整個系統的機會。在節點間重要訊息的傳遞上，使用密碼系統加以保密，確保整體系統的安全。當節點非正常模式離開系統的運作時，也能夠偵測到而移除，減少不必要的儲存節點資訊的成本。

5.2 未來發展與改進方向

雖然在實驗中用到了十餘台校內外真實的電腦主機參與測試，不過畢竟我們希望能處理的是真正廣大的網路使用者需求，在萬人以上的環境中依然能夠維持系統的運作和效率。所以希望系統整體完成時，可以由大量網路使用者真實使用來進行測試，對於一些參數和效能的測試，也才能調校出更佳的設置情形。

目前系統除了基本功能面大致完成外，仍有一些 error control 的部份尚未處理完全，另外也希望能夠加入 GUI 的部份，使得系統使用起來更人性化和方便。

此外，在網路節點管理的部份，若是有攻擊者節點滲入系統中，運行高效能的主機和網路，就很容易成爲群組的 Dynamic Subscribe Server，而輕易掌控群組中的 Slave Common Node，所以除了週期性更換 Dynamic Subscribe Server 之外，希望能制定出更好的策略來防止這種現象。

參考文獻

[1] 許志行，"網路匿名連線機制研究"，2005 年國立交通大學資訊科學所碩士論文

[2] 彭垂業，"在網路上的祕密通訊機制之研究"，2005 年國立交通大學資訊科學所碩士論文

[3] 彭垂業、許志行、楊武，"網路秘密通訊機制"，Proc. 2005 National Computer Symposium, Tainan, Taiwan, December 2005

[4] 陳冠志，"匿名網路之管理與連線分析"，2007 年國立交通大學資訊科學與工程所碩士論文

[5] Donald Welch, Senior Member, IEEE, and Scott Lathrop. "Wireless Security Threat Taxonomy", Proceedings of the 2003 IEEE Workshop on Information Assurance United States Military Academy, West Point, NY June 2003

[6] Michael G. Reed, Member, IEEE, Paul F. Syverson, and David M. Goldschlag. "Anonymous Connections and Onion Routing", Volume 16, Issue 4, May 1998 Page(s):482 – 494, Digital Object Identifier 10.1109/49.668972

[7] Nick Feamster, Roger Dingledine, "Location Diversity in Anonymity Network "

[8] Bruce Schneier, "Applied Cryptography, Second Edition, Protocols, Algorithms, and Source Code in C", John Wiley & Sons,Inc. 1996

[9] Paul A. Strassmann, William Marlow, "Risk-Free Access into the Global Information Infrastructure via Anonymous Re-Mailers", Symposium on the Global Information Infrastructure: Information, Policy & International Infrastructure Cambridge, MA, January 28-30, 1996

[10] Mixmaster Website, http://mixmaster.sourceforge.net/

[11] Danezis, G. Dingledine, R. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol", Security and Privacy, 2003. Proceedings. 2003

Symposium on Publication Date: 11-14 May 2003

[12] Mixminion Website, http://mixminion.net/

[13] Roger Dingledine, Nick Mathewson, Paul Syverson, "Tor: The Second-Generation Onion Router", Proceedings of the 13th conference on USENIX Security Symposium - Volume 13 table of contents, San Diego, CA, Pages: 21 – 21, 2004

[14] Tor Website, http://tor.eff.org/

[15] Michael J. Freedman, Robert Morris, "Tarzan: A Peer-to-Peer Anonymizing Network Layer", Proceedings of the 9th ACM Conference on Computer and Communications Security, November 2002.

[16] Tarzan Website, http://pdos.csail.mit.edu/tarzan/index.html

[17] Anonymizer Website, http://www.anonymizer.com/

[18] Weihan Wu, "深度學習 C++", http://www.math.ncu.edu.tw/~ziyou/c++/

[19] Wikipedia, http://en.wikipedia.org/wiki/Tor_(anonymity_network)

[20] A.J Ganesh, A.-M. Kermarrec, L Massoulie, "Peer-to-Peer Membership Management for Gossip-Based Protocols", Computers, IEEE Transactions on Volume 52, Issue 2, Feb. 2003 Page(s):139 – 149