行政院國家科學委員會補助專題研究計畫期末報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※ ※
※
※ **支援 IPv6 協定之快速位址搜尋方法研究設計** ※
※ Study and Design of Fast Address Lookup Scheme ※
※ Supporting IPv6 Protocol ※
※ ※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☑個別型計畫 整合型計畫

計畫編號：NSC－90－2213－E－009－167

執行期間：90 年 8 月 1 日至 91 年 7 月 31 日

計畫主持人：陳耀宗

計畫參與人員：詹益禎、郭國承、陳文彥

本成果報告包括以下應繳交之附件：

赴國外出差或研習心得報告一份

赴大陸地區出差或研習心得報告一份

出席國際學術會議心得報告及發表之論文各一份

國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊工程系所

中 華 民 國 九十一 年 十 月 十 日

「支援 IPv6 協定之快速位址搜尋方法研究設計」

Study and Design of Fast Address Lookup Scheme

Supporting IPv6 Protocol

計畫編號：NSC 90-2213-E-009-167
執行期限：91 年 8 月 1 日至 92 年 7 月 31 日
主持人：陳耀宗教授　國立交通大學資訊工程學系
計畫參與人員：詹益禎、郭國承、陳文彥

一、中文摘要

在路由器（Router）的設計中，如何快速並正確的將進入路由器的封包送到相對應的輸出埠（output port）是一件非常重要的工作。而決定封包屬於哪個輸出埠的這個程序，我們稱為位址搜尋。由於目前我們使用的 IPv4 封包格式與未來將應用於廣大網際網路的 IPv6 封包格式的長度不同，因此如何將原有的 IPv4 位址搜尋方法應用於 IPv6 的網路中，便成為一個值得探討的課題。

在這計畫中，我們提出一個可同時應用於 IPv4 與 IPv6 位址搜尋演算法。我們也利用軟體模擬的的方式加以驗證演算法的正確性。根據實驗的結果，證明該演算法有極佳的搜尋速度以及大幅減少執行時需要的儲存空間，並可實際應用於未來的路由器設計之中。

關鍵詞：位址搜尋，IPv6，分類器，封包分類，聚集，位元向量，過濾器衝突

英文摘要

For designing the router, it is an important task to distribute the incoming packets to their corresponding output port. The process that decides where an incoming packet should be forwarded is called IP address lookup. Because the formats of the current IPv4 and the emerging IPv6 are different, therefore how to apply the IPv4 address lookup schemes to the IPv6 protocol is a essential topic that needs investigation.

In this project, we propose an address lookup algorithm that could be applied to both IPv4 and IPv6 protocols. We also use the software simulation to validate the correctness of the algorithm. According to the experiment result, we prove that the proposed algorithm features excellent lookup speed and greatly reduces a large amount of memory requirement, and it could be deployed in the future router design.

Keywords: IP Lookup, IPv6, Classifier, Packet Classification, Aggregation, Bit Vector, Filter Conflict

二、緣由與目的

隨著網際網路的普及，每個人隨時隨地可以進入網際網路，搜尋自己需要的資訊。WWW（World Wide Web）的出現，也讓越來越多的使用者、電腦、以及網路參加了網際網路的聚會，因此網際網路上的訊務流量也日趨龐大。目前網際網路的訊務流量正以數個月倍增的驚人速度在增加。

由於使用者開始大量的使用網際網路的資源，使用者的需求也變的越來越多樣化，網路服務提供者勢必要因應使用者的需求調整其服務的品質，如此一來，路由器也必須提供一些像是 QoS（Quality of Service），虛擬私人網路（VPN），多重協定封包交換(MPLS)等的功能，以符合網路服務提供者的要求。上述幾項功能不能藉由單純的目的地位址搜尋來達成，所以

路由器必須擷取更多封包的資訊，例如：來源端位址(source address) 、來源端埠號(source port) 、目的端埠號(destination port) 、封包的協定等。這樣的一個方向其實是將原本位址搜尋的範圍擴大，將原本一個維度（dimension）的問題延伸成為多個維度，所以封包分類可以說是包含了位址搜尋的集合，探討這樣的問題時，可以完整涵蓋 IPv4 和 IPv6 的位址搜尋。

封包分類的動作基本上很簡單，不外乎擷取封包中的資訊，與資料庫中的各個欄位做比對，以找出最佳比對的過濾器，並採取所指定的動作。目前已有相當多的論文在探討這樣延伸的問題，它可以利用硬體的方式實做，也可利用軟體的方式進行模擬。在眾多方法中，令我們感興趣的是一個稱為朗訊位元向量（Lucent Bit Vector）的方式，這個方法原本是利用硬體的方式來探討封包分類的問題，但之後的一些論文則是開始利用軟體模擬的方式來研究該方法是否有改進的空間。由於此方法已經被路由器設計大廠朗訊採用，所以可以說是已被驗證可實際運用於目前的路由器中，如何改進這個方法便成為一個很有趣的問題。

三、研究方法與成果

在這一年的研究中，我們設計了一個新的封包分類演算法，它能有效的將進入路由器的封包分類至所屬的過濾器，並針對該過濾器指定的動作採取行動。我們著眼於朗訊位元向量方法耗費大量的儲存空間來維持位元向量(bit vector)，所以我們利用建立子樹(sub-tree)的觀念來節省儲存位元向量的空間，不同於朗訊位元向量的方法，我們僅在子樹的根節點(root node)儲存位元向量（Bit Vector），將此子樹含有位元向量的所有資訊都整合起來儲存於根節點的位元向量中，如此一來，位元向量所耗費的儲存空間會隨著子樹中所包含子節點個數的增多而節省其儲存空間。之外，由於我們觀察到只利用這樣單純的觀念並沒有完全解決封包分類的問題，而會導致所謂錯誤比對（false match）的情形發生。為了解決錯誤比對的情況，我們在事先處理過濾器資料庫（filter database）時，會將資料庫做分析，並將封包分類分離成兩個階段來處理，一個採取原本位元向量的方式，另一個則採取線性搜尋(linear search)的方法來比對。

以上的設計導因於我們的兩點觀察，第一，會發生錯誤比對的情形是由於我們採取子樹的概念來節省儲存空間，原本分屬於不同分支的資訊會因而整合到根節點中，使得在執行位元向量比對時找出錯誤的位置，所以我們在事先處理過濾器資料庫時，會把這些會發生錯誤比對的過濾器給分離成另一個資料庫，僅對於那些不會發生錯誤比對的產生位元向量，這樣可以得到的第一個優點是位元向量的長度大大減少，如此也是減少最差情形(worst case)的記憶體存取次數；第二，事實上，在過濾器資料庫含有的過濾器數量很少時，利用複雜方式的演算法所得到的效能並不會比簡單的線性搜尋方式來的快速。由於我們可以控制在第二階段要比對的過濾器數量大小，所以在第二階段利用線性搜尋的方式便可以達到極快的速度。以下即開始說明我們的方法是如何運作以及實驗的結果。

1. 我們的方法採用原本朗訊位元向量的方式來建立位元向量，其建立的方式是，若是過濾器資料庫中有 1,000 個過濾器，那麼我們的位元向量的大小便是 1,000 個位元(bit)，在我們的假設中，過濾器資料庫已經按照優先權(priority)的順序加以排序，越高優先權的過濾器放在資料庫的開始，相對應來說，越低位元位置所代表的便是越高的優先權，例如：位元 1 的優先權比位元 5 的優先權來的高。在進行位元向量的比對時，我們是利用交叉（intersection）的方式來比對，所以只要交叉比對出來的值等於 1，那麼該比對便可停止，此位元的位置也能直接對應到相對應的過濾器位址。

2. 為了節省儲存位元向量的位址空間，如前所述，我們利用建立子樹的觀念來建立位元向量，建立的方式如下：

```
GenerateSubtrieRoot(Cur_Node, Cur_Depth)
BEGIN
    IF ( ChildWithPrefix(Cur_Node) == Max_Prefix_Num )
        GenerateSubtrieRoot(Cur_Node);
    ELSE
        SubtrieConstructor(Cur_Node->left, Cur_Depth+1);
        SubtrieConstructor(Cur_Node->right, Cur_Depth+1);
        IF ( ChildWithPrefix(Cur_Node ) >=
        Max_Prefix_Num )
            GenerateSubtrieRoot(Cur_Node);
        ELSE IF ( (Cur_Node == Trie_Root) &&
                (ChildWithPrefix(Cur_Node) !=
                Max_Included_Prefix_Num) )
            GenerateSubtrieRoot(Current_Node);
END
```

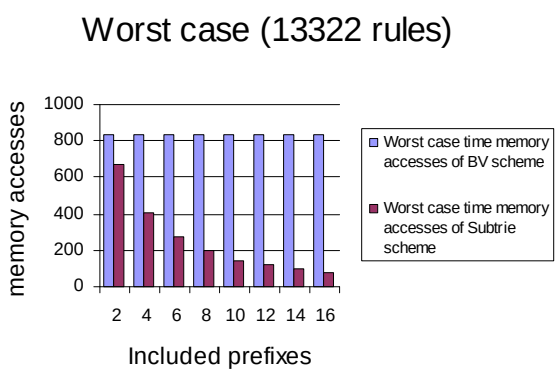
式，根據目前這個節點所包含的子樹中有多少個帶著 prefix 的節點以決定是否將此節點設成該子樹的根節點。藉由這樣的方式建構子樹的根節點，其目的是在有效的控制根節點維護的資訊量，也就是控制整合到根節點的位元向量的數目。整合的數目越多，位元向量中為 1 的位元便會增加，使得錯誤比對的機會大增，這個部分我們以第二階段的線性比對做為改進。

3. 建立完所有的根節點位置時，我們會給予每個根節點一個相對應的辨識碼，同時去修改過濾器資料庫，對於每個維度去更新相對應的辨識碼，接著搜尋整個過濾器資料庫，將所有維度辨識碼的集合相同之過濾器集中放置到第二階段處理，例如：以來源端位址的辨識碼為 2，目的端位址的辨識碼為 3，那麼我們便會去搜尋整個資料庫，將所有來源端位址的辨識碼是 2 和目的地位址的辨識碼是 3 的集合起來，放置到第二階段的資料庫。若是某個辨識碼的集合僅有自己而沒有其他人與其相同，那麼此過濾器一定不會與其他的過濾器產生錯誤比對的狀況，所以我們將此過濾器放置在第一階段處理。

4. 根據步驟 3 處理的結果，我們將第一階段的過濾器資料庫依照位元向量的方式來建構其位元向量，在這裡可以注意到的是因為我們在事先處理時已經將原本的過濾器資料庫分成兩個部分來考量，所以此時在第一個階段的位元向量比起朗訊位元向量會短小許多，這也可以視為我們的方法在基本上比朗訊位元向量的方式利用較少的記憶體存取次數。

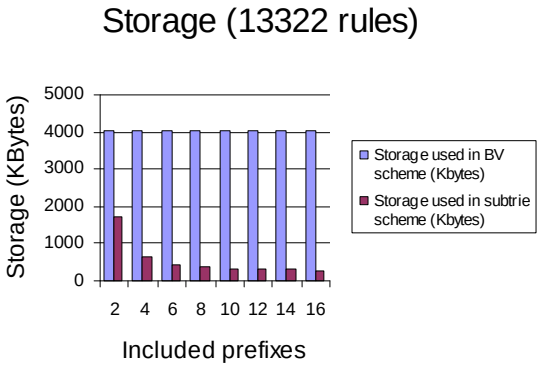
5. 在實際執行我們演算法的部分，如同原本朗訊位元向量的方式，我們會針對每個維度都進行位址搜尋的步驟，這個部分可以利用現有的 IPv4 以及 IPv6 快速位址搜尋資料結構，來加快此步驟的速度。接著，找出每個維度所符合的子樹根節點後，我們首先對每個維度的根節點做一次記憶體存取，讀出存於該節點的辨識碼，檢查該辨識碼的集合是否存在第二階段的辨識碼集合中，若是，則我們可直接跳過第一階段的比對，直接進入第二階段做處理，可以這麼做的原因是，由於我們已經先做了事先處理的步驟，所以可直接在這裡便直接決定是否跳躍進第二階段；若否，那麼代表該辨識碼不在第二階段中，所以我們可以很明確的決定進入第一階段的位元向量比對。

6. 下面兩個圖表是我們的實驗結果之一，在這個結果中，我們的過濾器資料庫是兩個維度的資料庫，包含來源端位址和目的端位址兩個欄位。實驗中，我們假設路由器一次記憶體存取能夠讀取出 32 個位元，所以對於朗訊位元向量的方式來說，其最差的記憶體存取次數是(過濾器個數/一次記憶體存取的位元數），結果如下圖：



可以發現我們方法的記憶體存取數量

會因為子樹包含的節點數量增多而變小。在這裡我們會對最差的狀況有興趣的原因是，由於過濾器資料庫的型態會根據網路的特性而改變，以致於若探討平均的記憶體存取數時，其變動的因素遠大於最差的記憶體存取數，所以若是能提供一個在最差狀況下能夠表現優良的封包分類演算法，代表的是該演算法在平均的狀況下也會有不錯的效能。下圖是我們演算法所耗費的儲存空間，可以清楚的發現依然遠小於朗訊位元向量的儲存空間。



四、結論與討論

為了符合使用者日益變化的需求，網路服務提供者必須適應使用者的要求提供服務，因此在路由器的設計中也必須考量到更多的層面，而不僅是侷限於單一的目的地位址搜尋。解析封包更多的欄位進行分析，快速決定封包所屬的過濾器並針對封包採取動作，是未來路由器不可或缺的功能。所以如何應用快速，正確的多欄位封包分類演算法，將會是未來路由器研發設計時一個重要的考量。

依據本計畫在過去一年之研究，我們提出一個可實際使用在未來 IPv6 路由器設計的演算法，藉由實驗結果的呈現，得知我們可以依據實際網路的需求和路由器本身的硬體限制，調整演算法以得到最佳的效能，同時適應各種不同的分類器特性，這是與其他方法最大的不同之處。我們依照預定之預算，完成 100％預期之研究目標，此外，在本計畫研究過程的經驗，我們發現其它值得探討的研究方向，可做為未來研究計畫的參考。

五、參考文獻

[1] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, “Fast and Scalable Layer Four Switching,” Proc. ACM SIGCOMM ‘98.

[2] M. Waldvogel, G. Varghese, “Scalable High Speed IP Routing Lookups,” Proc. ACM SIGCOMM ’97, Sept. 1997, pp. 25-36.

[3] B. Lampson, V. Srinivasan, and G. Varghese, “IP Lookups Using Multiway and Multicolumn Search,” Proc. IEEE INFOCOM ’98, Apr. 1998, pp.1248-56.

[4] T. V. Lakshman and D. Stidifialis, “High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching,” Proc. ACM SIGCOMM ’98, Sept. 1998.

[5] A. Feldman and S. Muthukrishnan, “Tradeoffs for Packet Classification,” Proc. IEEE INFOCOM ’00, Mar. 2000, pp.397-413.

[6] P. Gupta and N. McKeown, “Packet Classification on Multiple Fields,” Proc. ACM SIGCOMM ’99, Sept. 1999.

[7] F. Baboescu and G. Varghese, “Scalable Packet Classification,” Proc. ACM SIGCOMM ’01, Aug. 2001.

[8] Ji Li, Haiyang Liu, and Karen Sollins, “Scalable Packet Classification Using Bit Vector Aggregating and Folding,” Proc. ACM SIGCOMM ’02, Aug. 2002.

[9] V. Srinivasan, S. Suri, and G. Varghese, “Packet Classification using Tuple Space Search,” Proc. ACM SIGCOMM ’99.

[10] P. Gupta and McKeown, “Classification Using Hierarchical Intelligent Cuttings,” Proc. Hot Interconnects VII, Aug. 1999.;also available in IEEE Micro, vol. 20. no. 1, Jan./Feb. 2000, pp. 34-41.

[11] Abilene NetFlow Nightly Reports, http://www.itec.oar.net/abilene-netflow/