IP Subnetting: 原理與觀察

黃俊穎 林盈達

高速網路實驗室@交通大學

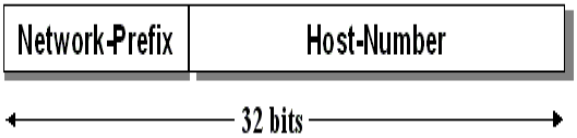
gis89501@cis.nctu.edu.tw ydlin@cis.nctu.edu.tw

摘要

當我們在區域網路下設定讓自己的電腦上網時，若不使用自動取得 IP 地址的機制，那麼就得填入如 IP 地址、Netmask、DNS 等相關資訊。本文解釋 IP 地址與 Netmask 在 host 端及 router 端的運作方式，包括發送封包、選擇路徑以及區域廣播，讓大家對網際網路的底層技術有更深入一層的了解。

1. 背景

傳統上我們把 Internet 上 32-bit 的 IP 位址分爲二大部份，前面一部份爲 Network Prefix，後面一部份爲 Host Number，如圖一所示[1]。其中，Host-Number 全爲 0 表示該 Network 的 Network ID；Host-Number 全爲 1 表示該 Network 的廣播位址。



圖一：傳統 IP 位址配置方式

爲了不同的使用量需求，根據 Network-Prefix 的長度又分爲三大類，分別爲 Class A、Class B 和 Class C。如表一所示。

表一：三大類 IP 位址分佈情形

Class	Leading Bits	Prefix Length	Range	Netmask
A	0	8	0.0.0.0-127.255.255.255	255.0.0.0
B	10	16	128.0.0.0-191.255.255.255	255.255.0.0
C	110	24	192.0.0.0-239.255.255.255	255.255.255.0

從表一中我們可以看出這三大類 IP 地址的分佈情形和數量。以 Class B 爲例，Network Prefix 的長度爲 16-bit，而其前二個 bit 是 1 和 0，因此 Class B 就一共有 2^{16-2} 這麼多個 Networks，而每個 Network 之下就有 2^{32-16} 個 IP 位址可用。

在這種使用方式之下便延伸出一些問題。我們可以發現不同 Class 的 Network 之下所分配到的可使用 IP 量差異實在是很大，而在同一個 Network 中的廣播流量也大的驚人。試想在 Class A 的 Network 中，某一個 IP 位址做廣播就會讓這個 Network 中的其他數以萬計的 IP 位址也收

到，如果有好幾個 IP 位址都在廣播，那這個區段的網路可能就癱瘓了。此外，另一個嚴重的問題就是 Router 上的路徑表會成長的十分龐大，以應付如此多的 Networks 及 Hosts。

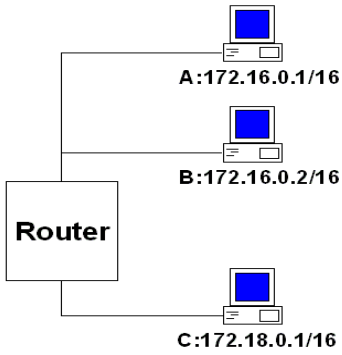
爲了解決這個問題，於是便有「子網路(Subnet)分割」的想法被提出。本文所介紹的，即是目前廣爲使用的網路分割方式以及 Host、Router 端對 Netmask 的運作和處理方式。

2. Netmask 的運作方式[2]

2.1 Host 端

我們首先得了解 Netmask 的運作方式。在 Host 端方面，如圖二所示，這是二個 Class B 的 Netowk。當 A 要送封包給同一個 Network 內的 B 時，便執行下列運算：

1. 拿自己的 IP 地址和自己的 Netmask 做 AND 運算: 172.16.0.1 & 255.255.0.0 == 172.16.0.0。
2. 拿 B 的 IP 地址和自己的 Netmask 做 AND 運算: 172.16.0.2 & 255.255.0.0 == 172.16.0.0。
3. 比較 1、2 的結果，發現結果相同，因此 A 便將封包直接傳送給 B。



圖二：網路示意圖

當 A 是要將封包送給不同 Network 之下的 C 時，同樣執行上述的運算：

1. 拿自己的 IP 地址和自己的 Netmask 做 AND 運算: 172.16.0.1 & 255.255.0.0 == 172.16.0.0。
2. 拿 C 的 IP 地址和自己的 Netmask 做 AND 運算: 172.18.0.1 & 255.255.0.0 == 172.18.0.0。
3. 比較 1、2 的結果，發現結果不同，因此 A 將封包直接傳送給 Router，由 Router 轉送給 C。

這裡所謂「直接傳送」封包在 Ethernet 上即由發送端以 ARP 協定取得對方的 Ethernet Address 之後，再往該位置傳送。

2.2 廣播封包

廣播封包的範圍也是用 netmask 來定義的。假設我們從 172.16.1.1/16 這個設定送出廣播封包，那麼其廣播的位址爲 172.16.255.255，而只有在 172.16.0.0 這個 network 之下的機器會收到這個廣播封包。

2.3 Router 端

在 Router 端，一樣得拿 netmask 來運算，以得知該將封包往哪裡送，這裡的查詢路由表動作，須進行"Longest Prefix Match"

表二：Routing Table 範例

Entry	Destination	Netmask	Gateway
1	172.16.0.0	255.255.0.0	gw1
2	172.16.1.0	255.255.255.0	gw2
3	172.16.2.0	255.255.255.0	gw3

以表二爲例，當 Router 收到一個要送往 172.16.2.1 的封包時，所需的動作如下：

1. 比較第一筆紀錄: 172.16.2.1 AND 255.255.0.0 == 172.16.0.0；
   172.16.0.0 AND 255.255.0.0 == 172.16.0.0，結果相符合。
2. 比較第二筆紀錄: 172.16.2.1 AND 255.255.255.0 == 172.16.2.0；
   172.16.1.0 AND 255.255.255.0 == 172.16.1.0，結果不符合。
3. 比較第三筆紀錄: 172.16.2.1 AND 255.255.255.0 == 172.16.2.0；
   172.16.2.0 AND 255.255.255.0 == 172.16.2.0，結果相符合。
4. 第一筆和第三筆紀錄皆符合，但第三筆的 netmask 有 24-bit 較長，因此選擇往 gw3 送封包。

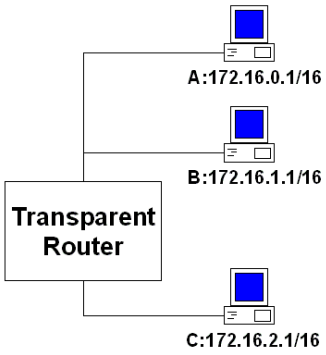
3. 分割子網路的方式

了解了 netmask 的運作方式後，接下來我們來看看常用的子網路分割法有哪些。在這裡我們介紹二種方式，一種是 transparent router，另一種方式就是利用不同長度的 netmask。

3.1 使用 Transparent Router

一般而言，若我們要傳送封包的目的地是屬於相同的子網路，那麼在傳送之前，我們得先利用 ARP 取得目的地 IP 地址相對應界面的 Ethernet Address，然後才能將封包送往目的地。

利用這個概念，在一個大的網路環境裡，將其分割成數塊，每一塊以 transparent router 連接即可。如圖三。



圖三：TR

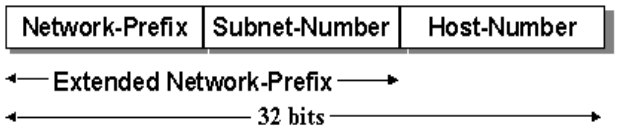
如果是由 A 送封包到 B，那麼流程還是和之前一般 Router 的情況相同。不過，如果是從 A 送到 C，經過 A 運算後，A 發現自己和 C 是屬於同一個 network，於是便直接將封包直接往 C 送。但實際上，A 和 C 是在不同的實體網路之下，因此當 Transparent Router 收

到由 A 發出的 ARP Request 之後，發現 C 和 A 是位於不同邊，便代 C 回答，讓 A 把封包送過來後，再將封包轉送到 C。反之，C 送給 A 的時候亦同。

Transparent router 常以 proxy ARP 的方式實作。即 router 上紀錄其下各個子網域下的 Ethernet address，並將這些資訊共用於連接其他子網路的界面，如此就可以讓連接於 router 上的幾個子網路透過這一個 router 而相互連結。

3.2 使用不特定長度的 Netmask

這個方法的原理就是從原來的三大類 IP 裡的 Host-Number 裡再取幾個 bits 出來做「Subnet-Number」，如圖四所示。



圖四：分割子網路後的 IP 配置方式

Subnet-Number 這一部份可以是固定長度或是不固定長度。以固定長度爲例，我們可以在一個 Class B 的 IP 位址下再拿 2 個 bits 出來做 Subnet-Number，如此就可以分割出四個子網路。每個子網路所使用 Netmask 的長度就變爲 18 bits (255.255.192.0)，如表三所示。

表三：固定 Netmask 長度的切法

Subnet Number	Least-Significant 2 bytes	Address	# of IP Address
140.113.0.0	00000000.00000000	140.113.0.0-140.113.63.255	2^{14}
140.113.64.0	01000000.00000000	140.113.64.0-140.113.127.255	2^{14}
140.113.128.0	10000000.00000000	140.113.128.0-140.113.191.255	2^{14}
140.113.192.0	11000000.00000000	140.113.192.0-140.113.255.255	2^{14}

再以不固定長度的 Netmask 爲例，我們一樣在一個原本爲 Class B 的 IP 地址之下將其切割爲四個子網路，由於每個子網路的 Netmask 不同，因此每個子網路之下所能容納的 IP 數目就不同，如表四所示。

表四：不固定 Netmask 長度的切法

Subnet Number	Least-Significant 2 bytes	Address	# of IPs
140.113.0.0	00000000.00000000	140.113.0.0-140.113.127.255	2^{15}
140.113.128.0	10000000.00000000	140.113.128.0-140.113.191.255	2^{14}
140.113.192.0	11000000.00000000	140.113.192.0-140.113.223.255	2^{13}
140.113.224.0	11100000.00000000	140.113.224.0-140.113.255.255	2^{13}

其中，每個子網路的 Host-Number 部份，若 bits 全爲 0 表示該子網路的 Network ID；若 bits 全爲 1 表示該子網路的廣播位址。

使用 netmask 來分割子網路是目前最常用的方式，一般的 Router 只要能處理不特定長度的 netmask 就能正確的選擇封包的路徑，而 host 端只要遵照原本發送封包的運算方式即可。

4. 實際觀察

接下來，我們分別從 host 及 router 端上來觀察 netmask 的運作情形。在 host 端方面，我們以 ping 和 tcpdump 觀察；在 router 端方面，我們以分析 Linux 核心原始碼在 routing table look up 和 routing table cache。

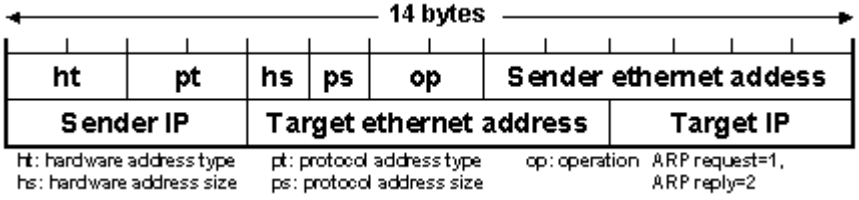
4.1 觀察封包的流向

有了上述的基本概念之後，我們可以在 host 端做一些簡單的觀察，了解 netmask 在 host 端的運作。我們使用 ping 及 arp 指令配合 tcpdump 這個小工具，觀察封包的收發及 ARP cache 的內容以了解 netmask 的作用。

我們先觀察以 netmask 分割一個 Class B 子網路透過 router 轉送的情形。發送端首先把自己的 ARP cache 清空，然後分別用 ping 送一個封包給同一個子網路裡的機器、預設閘道以及不同子網路裡的機器，然後再觀察 ARP cache 裡的內容。同時全程以 tcpdump 監看封包。實驗用的主機 IP 爲 140.113.88.145，預設閘道則是 140.113.88.254。

```
$ ping –c 1 140.113.88.186 > /dev/null ..... ①
$ arp –a
SpeedPC186.cis.nctu.edu.tw (140.113.88.186) at 00:90:27:36:4F:3B [ether] on eth0
$ ping –c 1 140.113.179.73 > /dev/null ..... ②
$ arp –a
SpeedPC186.cis.nctu.edu.tw (140.113.88.186) at 00:90:27:36:4F:3B [ether] on eth0
? (140.113.88.254) at 00:20:9C:06:E1:FE [ether] on eth0
```

我們可以發現 ping 同一個子網路裡的機器時，由 ARP 所取



圖五：ARP 封包格式

得的就是該機器網路界面的 Ethernet address；而 ping 不同子網路裡的機器時，由 ARP 所取得的則是預設閘道的 Ethernet Address。Host 端使用 netmask 計算後，判斷目的地是否屬於同一個子網路，以決定該把封包送往哪裡。我們可以從 tcpdump[3] 截取到的 ARP 封包看出。ARP 的封包格式如圖五所示。

```
① 23:34:43.199869  arp who-has SpeedPC186.cis.nctu.edu.tw tell 140.113.88.145
                   0001 0800 06 04 0001 0080 c88c 39d1 8c71 5891 0000 0000 0000 8c71 58ba
   23:34:43.200228  arp reply SpeedPC186.cis.nctu.edu.tw is-at 0:90:27:36:4f:3b
                   0001 0800 06 04 0002 0090 2736 4f3b 8c71 58ba 0080 c88c 39d1 8c71 5891
② 23:35:27.180271  arp who-has 140.113.88.254 tell 140.113.88.145
                   0001 0800 06 04 0001 0080 c88c 39d1 8c71 5891 0000 0000 0000 8c71 58fe
   23:35:27.181339  arp reply 140.113.88.254 is-at 0:20:9c:06:e1:fe
                   0001 0800 0604 0002 0020 9c06 e1fe 8c71 58fe 0080 c88c 39d1 8c71 5891
```

從上面的結果②我們可以看出，當送封包給不同子網路的 IP 位址時，便直接送出 ARP 封包詢問預設閘道的 Ethernet Address 再將封包送給他處理。接下來，我們再觀察以 transparent router 分割 Class B 子網路的情形。我們一樣以 ping、arp 以及 tcpdump 來觀察，實驗用的主機 IP 不變，不過 netmask 設定改爲 255.255.0.0，即傳統 Class B 的 netmask。這次除了 ping 目的機器外，也 ping 原來的預設閘道。

```
$ ping –c 1 140.113.179.73 > /dev/null ..... ③
$ arp –a
ngi13.eic.nctu.edu.tw (140.113.179.73) at 00:20:9C:06:E1:FE [ether] on eth0
$ ping –c 1 140.113.88.254 > /dev/null ..... ④
$ arp –a
? (140.113.88.254) at 00:20:9C:06:E1:FE [ether] on eth0
ngi13.eic.nctu.edu.tw (140.113.179.73) at 00:20:9C:06:E1:FE [ether] on eth0
```

我們可以發現預設閘道與目的地機器的 Ethernet address 是相同的。這是因爲雖然從 IP 和 netmask 上來看這二個位址是屬於同一個子網路，但實際上，這是被切成二塊的子網路，所以當 host 端查詢 ARP 位址時，由居中的 transparent router 代爲回答，因此得到的 Ethernet address 位址就是相同的。我們可以從 tcpdump 的結果看出在送封包時，host 端直接詢問目的地 IP 的 Ethernet address，而由居中的 transparent router 回答其 Ethernet address，讓封包送交 transparent router 處理(如③之 ARP 結果所示)，以正確到達目的地。

```
③ 23:54:39.751972  arp who-has ngi13.eic.nctu.edu.tw tell 140.113.88.145
                   0001 0800 0604 0001 0080 c88c 39d1 8c71 5891 0000 0000 0000 8c71 b349
   23:54:40.118787  arp reply ngi13.eic.nctu.edu.tw is-at 0:20:9c:06:e1:fe
                   0001 0800 0604 0002 0020 9c06 e1fe 8c71 b349 0080 c88c 39d1 8c71 5891
④ 23:55:33.724277  arp who-has 140.113.88.254 tell 140.113.88.145
                   0001 0800 0604 0001 0080 c88c 39d1 8c71 5891 0000 0000 0000 8c71 58fe
   23:55:33.725466  arp reply 140.113.88.254 is-at 0:20:9c:06:e1:fe
                   0001 0800 0604 0002 0020 9c06 e1fe 8c71 58fe 0080 c88c 39d1 8c71 5891
```

4.2 廣播封包的處理

如果我們用 ping 及 tcpdump 的方式來觀察當廣播封包被送出的情形，我們會發現我們沒有先送出 ARP-request 就直接對廣播位址送出 ICMP-request，接下來才是由同一個 network 之下的其他機器做 ARP-request 並回送 ICMP-reply 封包。

```
# tcpdump –n arp or icmp
tcpdump: listening on eth0
17:16:20.434312 140.113.179.73 > 140.113.179.255: icmp: echo request
17:16:20.434567 arp who-has 140.113.179.73 tell 140.113.179.250
17:16:20.434645 arp reply 140.113.179.73 is-at 0:40:5:30:8e:33
17:16:20.434601 arp who-has 140.113.179.73 tell 140.113.179.180
17:16:20.434680 arp reply 140.113.179.73 is-at 0:40:5:30:8e:33
17:16:20.434698 arp who-has 140.113.179.73 tell 140.113.179.59
17:16:20.434721 arp reply 140.113.179.73 is-at 0:40:5:30:8e:33
17:16:20.434768 140.113.179.250 > 140.113.179.73: icmp: echo reply
17:16:20.434952 140.113.179.180 > 140.113.179.73: icmp: echo reply
17:16:20.435089 140.113.179.59 > 140.113.179.73: icmp: echo reply (DF)
17:16:20.437346 140.113.179.254 > 140.113.179.73: icmp: echo reply
17:16:20.445248 arp who-has 140.113.179.73 tell 140.113.179.71
17:16:20.445285 arp reply 140.113.179.73 is-at 0:40:5:30:8e:33
17:16:20.446197 140.113.179.71 > 140.113.179.73: icmp: echo reply
……
```

如果去看 Linux 核心原始碼[4]，我們可以發現在查詢 Ethernet ARP cache 之前，如果發現目的地的 IP 位址是一些特別的位址，如廣播位址或是安裝在自己機器上的網路卡 Ethernet Address 等，那麼作業系統核心就會直接把特定的 Ethernet Address 填入而不做 ARP-request 的動作。以廣播封包而言，被填入的 Ethernet Address 就是 ff:ff:ff:ff:ff:ff。而這個廣播封包只有在同一個實體網路之下的機器收得到。相關的程式碼如程式碼列表一所示。

```
01:static int arp_set_predefined(int addr_hint, unsigned char * haddr, u32 paddr,
     struct device * dev)
02:{
03:   switch (addr_hint) {
04:   case RTN_LOCAL:
05:         printk(KERN_DEBUG "ARP: arp called for own IP address\n");
06:         memcpy(haddr, dev->dev_addr, dev->addr_len);
07:         return 1;
```

```
08:   case RTN_MULTICAST:
09:         arp_mc_map(paddr, haddr, dev, 1);
10:         return 1;
11:   case RTN_BROADCAST:
12:         memcpy(haddr, dev->broadcast, dev->addr_len);
13:         return 1;
14:   }
15:   return 0;
16:}
17:
18int arp_find(unsigned char *haddr, struct sk_buff *skb)
19:{
20:   ……
21:   if (arp_set_predefined(inet_addr_type(paddr), haddr, paddr, dev))
22:         return 0;
23:   ……
24:}
```

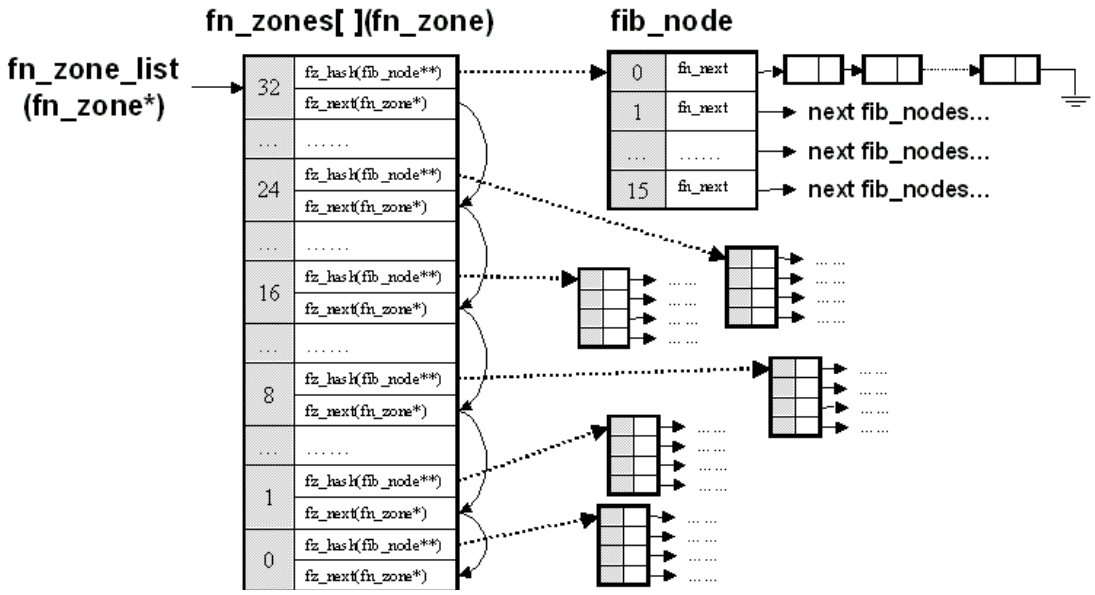
程式碼列表一：linux/net/ipv4/arp.c

其中 inet_addr_type(linux/net/ipv4/fib_frontend.c) 函式傳回的即是 IP 目的地位址的型態是屬於 BROADCAST、MULTICAST 或是 UNICAST。

4.3 Routing table lookup

當封包第一次進入作業系統核心要求轉送時，作業系統核心便會根據封包的目的地決定要將封包往哪個路徑送。而選擇路徑時則根據各個路徑上所設定的 netmask 配合 longest prefix match 原則選擇最適合的路徑。

Linux 核心以二層 hash table 配合 link list 來實作，如圖六所示。第一層 hash 是以 netmask 的長度來決定；第二層 hash 則是以封包的目的地來決定。在查詢時，從 netmask 較長的紀錄開始查詢，因此只要一查到符合目的地的資料，那麼查到的資料一定就會是符合且 netmask 最長的資料。



圖六：Linux kernel routing table 資料結構示意圖

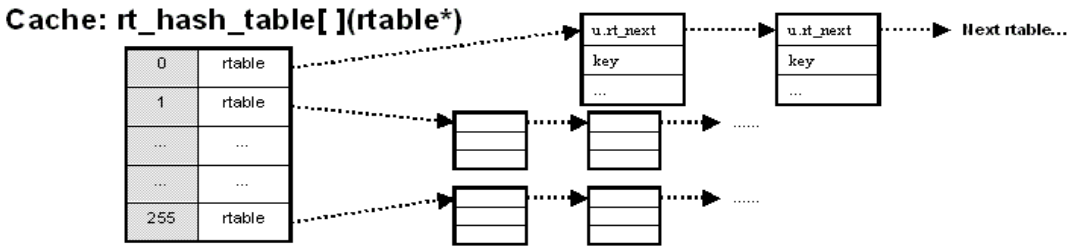
相關的查詢程式碼如程式碼列表二所示：

```
01: static int fn_hash_lookup
02:   (struct fib_table *tb, const struct rt_key *key, struct fib_result *res) {
03:   struct fn_zone *fz;
04:   struct fn_hash *t = (struct fn_hash*)tb->tb_data;
05:   for (fz = t->fn_zone_list; fz; fz = fz->fz_next)   {      /* For each zone */
06:         struct fib_node *f;
07:         fn_key_t k = fz_key(key->dst, fz);
08:         for (f = fz_chain(k, fz); f; f = f->fn_next)   {      /* For each node */
09:               if (matched)
10:                     fill result and return(0);
11:         }
12:   }
13:   return 1;
14: }
```

程式碼列表二：linux/net/ipv4/fib_hash.c

當查詢到適當的路徑後，Linux 核心會把查到的路徑給 cache 起來，以方便日後使用。同時，爲了加速往後處理封包的速度，處理封包的函式(如 forward、local deliver 或是 multicast)也會一起被紀錄起來。

4.4 Routing table cache



圖七：Linux kernel routing cache 資料結構示意圖

一但作業系統核心轉送過任何一個封包後，便會將這次查詢的結果放到 routing cache 紀錄中，下次還有封包要到同一個地方，如果目的地相同的話，那就不需要再到龐大的 routing table 裡重新尋找。因此，在真正開始查詢 routing table 之前，作業系統核心會先查詢 routing cache 裡是否已經有紀錄了。

Routing cache 的資料結構比 routing table 單純，他是一個有 255 個 bucket 的 hash table，每一個 bucket 是一個 link list，如圖七所示。在查詢時，只要先做一次 hash 的動作，然後對 link list 做 linear search 即可。相關的程式碼如程式碼列表三所示。

```
01: int ip_route_input(struct sk_buff *skb, u32 daddr, u32 saddr, u8 tos,
      struct device *dev) {
02:   ……
03:   hash = rt_hash_code(daddr, saddr^(oif<<5), tos);
04:   for (rth=rt_hash_table[hash]; rth; rth=rth->u.rt_next) {          /* cache lookup */
05:         if (rth->key matched) {
06:               fill result and return(0);
07:         }
08:   }
09:   ……
10:   return(ip_route_input_slow(skb, daddr, saddr, tos, dev));
11: }
```

程式碼列表三：linux/net/ipv4/route.c

5. 結論

子網路分割的技術在 Internet 上已使用多年，對網路使用者或是系統管理者而言，正確的設定 netmask 才能讓自己所使用的網路環境正常的運作。我們亦可以利用 netmask 來適當的分割網路，提高網路的使用率及效能。了解這個基本的原理後，日後在網路疑難問題排解上應該會更加的得心應手。

6. 參考資料

[1] Thomas A. Maufer, “IP Fundamentals”, PTR PH, 1999

[2] Ying-Dar Lin, “Classic Internet Protocols”(course slides), http://speed.cis.nctu.edu.tw/~ydlin/course/cn/part2/classicip.pdf, NCTU, May 1999.

[3] tcpdump-3.4, http://www.tcpdump.org/, Jan 1998

[4] “Linux Kernel 2.2.13”, http://www.kernel.org/,Oct 1999.