High-Speed Route Search Mechanisms for IPv6*

Wen-Shyen E. Chen†, Chung-Ting Justine Tsai, Lu-Meng Hsu, and Chih-Heng Bryan Wu
Institute of Computer Science
National Chung-Hsing University
Taichung, Taiwan, ROC
{echen,cttsai}@cs.nchu.edu.tw

Abstract

The Internet's astounding growth has begun to stress the technology that supports it, namely, the current Internet Protocol Suite, IPv4. IPv6 (Internet Protocol, Version 6) was developed to address the issues. The IPv6 resolves the Internet scaling problem, provides a flexible transition mechanism, and meets the needs of such new market as nomadic personal computing devices, networked entertainment, and device control. Nevertheless, the specification of IPv6 does not provide mechanisms to speed up the processing of the packets in the routers. As a result, the introduction of IPv6 might demand more computation from the already overloaded routers.

In this paper, we propose a high speed IP lookup scheme for IPv4 and then describe how the mechanism can be scaled to IPv6 in order to speed up the packet processing in the routers. The proposed mechanisms search for the Best Matching Prefix (BMP) by using *forwarding tables* consisting of Prefix Information Tables (PITs) and Lookup Tables (LTs) that provide guidelines for efficient search. The FPGA implementation of the scheme for IPv6 is also shown to validate its feasibility. For IP lookup in IPv4, the scheme needs 1 memory access in the best case, and 2 memory accesses plus one hash in the worst case to locate the BMP. It requires only 560 KBytes to 670 KBytes of memory space when about 45,000 routing tables entries in the backbone are logged for simulation. More memory (about 300 Kbytes more) can be used to reduce the worst case access time to 2 memory accesses. When 50ns DRAM is used for the forwarding tables, the scheme offers lookup speed of 10 millions packets per second The lookup speed can be improved linearly with the speedup of the type of memory used. Further, it does not need complex compression mechanisms to reduce the memory requirement. Although the currently available traces and routing entries for IPv6 do not provide the scale similar to that of the IPv4, initial analysis shows the approach takes at most 3 hashes and 2 memory accesses to locate the BMP in IPv6. Parallelism can be explored to further improve the lookup performance.

1 Introduction

An IP lookup scheme is the most fundamental operation in any IP routing product. A packet is received with a specific IP Destination Address (DA), a unique 32-bit field in current IPv4 and a 128-bit field in IPv6. The router must search *forwarding tables*[1] using the DA as its key, and determine which entry in the table represents the best route for the packet to its destination. (It is used for IP→ Output Port mapping when IP Lookup is performed.) The search is complicated by the fact that entries in the table have variable lengths, and also that many entries may represent valid routes to the same destination. This was done to conserve the memory space used by the forwarding tables. Unlike a simple search or hash that seeks to find an *exact match* within a table, the IP lookup algorithm must select the most specific route from a number of entries (known as the Best Matching Prefix, BMP) for the given DA.

The problem has become more challenging as data links now operate routinely at 100 MBits/second, and generate nearly 150,000 PPS that need to be routed. A lot of people have the illusion that the Internet Protocol, Version 6 (IPv6) [1] provides a solution for the performance issues. Although IPv6 resolves the Internet scaling problem and provides a flexible transition mechanism, the specification of IPv6 does not specify mechanisms to speed up the processing of the packets in the routers. As a result, the introduction of IPv6 might

*This work is supported in part by the National Science Council, ROC, under contract number NSC 89-2213-E-005-017.

†Corresponding Author. Email: echen@cs.nchu.edu.tw Phone:+886-4-285-3141 Fax: +886-4-285-3869

[1] In this paper, we use *routing table* to refer to the table created and updated by the routing processes, while *forwarding tables* are the tables that are referenced when IP Lookup is performed. These two are specifically separated as in many high performance multilayer switches, each port has its own lookup tables. The lookup tables are then updated periodically by the routing table used by the routing processes.

demand more computation from the already overloaded routers.

In order to meet the demands for next generation high-speed routers, IP lookup schemes that are efficient, flexible, and cost-effective are required.

In this paper, we propose a high-speed IP lookup scheme for the BMP by using *forwarding tables* consisting of Prefix Information Tables (PITs) and Lookup Tables (LTs) that provide guidelines for efficient search. The scheme scales very well as the sizes of the address and the routing table increase. For IP lookup in IPv4, the scheme needs 1 memory access in the best case, and 2 memory accesses plus one hash in the worst case to locate the BMP. The scheme needs only 560 KBytes to 670 KBytes of memory space when about 45,000 routing tables entries in the backbone are logged for simulation. More memory (about 300 Kbytes more) can be used to reduce the worst case access time to 2 memory accesses. If 50ns DRAM is used for the forwarding tables, the scheme offers lookup speed of 10 millions packets per second (10 MPPS). The lookup speed can be improved linearly with the speedup of the memory used. (So if 5ns SRAM is used, the lookup speed can be 100 MPPS.) Further, it does not need complex compression mechanisms to reduce the memory requirement. Its operation is simple, lending itself for easy hardware implementation. Although updates to the routing table degrade the lookup performance, the effect is found by simulation to be minimum. Compared to the related work in the literature, the scheme provides an efficient use of memory and at the same time, maintains the high-speed lookup required by the next generation routers. The detail of how the proposed scheme can be extended for IPv6 is also described. Although the currently available traces and routing entries for IPv6 do not provide the scale similar to that of the IPv4, initial analysis shows the approach takes at most 3 hashes and 1 memory access to locate the BMP in IPv6. Parallelism can be explored to further improve the lookup performance. Therefore it can be used to speed up the packet processing in the next generation IP routers.

There are many IP lookup schemes proposed in the literature [2, 3, 4, 5, 6, 7, 8, 9, 10]. Due to space limitation, the overview of the approaches is omitted here. Please refer to the individual work for the description of the schemes.

The rest of the paper is organized as follows. Section 2 describes the proposed high-speed IP lookup scheme. The performance evaluation, including the sustainable lookup speed and memory requirement, for the proposed IPv4 lookup scheme is done in Section 3 by simulation using the traces from the Internet backbone. The results obtained from the related work and the proposed scheme are then compared. Section 4 extends the basic scheme to IPv6. Hardware implementation (in FPGA) is also described to show its feasibility. Finally, concluding remarks are made in Section 5.

2 The proposed lookup scheme for IPv4

Although the aforementioned related works all have their advantages, however, the approaches either use complicated data structures, resulting in increased complexity while updating the routing entries, such as [10], or they are not scalable to IPv6, such as [5]. We propose a scalable IP lookup mechanism that is memory efficient and supports incremental updates to the lookup table when routing entries are modified. The idea behind the proposed scheme is described as follows.

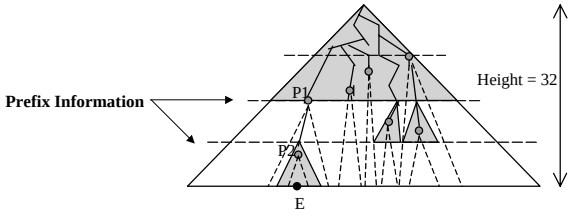

Figure 1: Basic idea of the proposed scheme.

By visualizing the Trie corresponding to the routing table, the nodes in the Trie representing the entries in the routing table scatter throughout the Trie. The basic concept behind our proposed scheme is that while doing the search for the longest prefix matching entry for the DA, we should quickly identify the approximate location of the entry first by a binary search based on the length of the prefix entries, then do a more complete search within a much smaller area to reduce the time spent in locating the entry. The approach is depicted in Fig. 1.

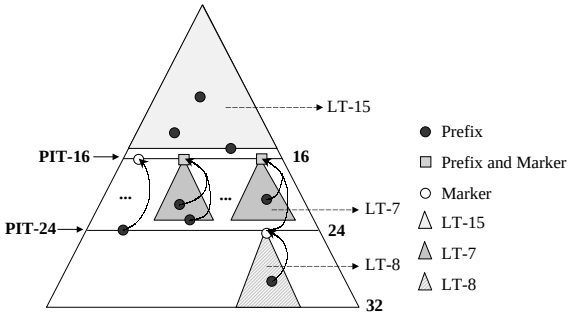

Figure 2: Tree structure in the Basic Scheme.

2.1 Terminologies and data structures used

Before introducing the proposed scheme, we first define some consistent terminologies that will be used throughout the text.

Terminologies:

- **Level**: In the tree, let the root be at level 0 initially. For all subsequent nodes, the level is the level of the node's parent plus one.

- **Height or Depth**: the maximum level of any node in the tree.

- **Subtree(s, l, h)**: a set in the tree rooted s at level l, and the height of the subtree i

- **Leaves_Subtree(s, l, h)**: leaves of the Su l, h).

- **Prefix($a_1 \rightarrow a_m$, n)**: first n MSB of the $a_1 \rightarrow a_m$, which is usually abbreviated tc a_m/n), meaning the route address $a_1 \rightarrow$ the prefix length of n.

- **LT-n**: a fully-expanded lookup table (descr low) for the Leaves_Subtree(s, l, n), which a mapping between the leaf nodes to thei sponding output ports.

- **LT(s, l, h)**: a fully-expanded lookup table Leaves_Subtree(s, l, h), which provides a between the leaf nodes to their correspond put ports.

- **PIT-n**: the Prefix Information Table (d below) corresponds to level n in the trie.

With the concept in mind, two data struct used to aid the lookup process, as can be se Fig. 2. The first one is the *Prefix Informatic* (PIT) that corresponds to a level in the Trie. PIT-n to represent the PIT at level n. The PIT the prefix information and provides information the binary search for the BMP. By dividing the Trie into small number of regions, we can effectively reduce the search time. However, some mechanism, such as the PITs in our scheme, needs to be in place to conserve the memory space used.

An entry in PIT-l indicates that there exists at least one node in the Subtree(s, l, 32-l) with the entry itself as a root. Note that multiple nodes in the subtree can be represented by a single entry in the PIT. The PITs are used as a basis for binary search. With our approach, only two PITs, PIT-16 and PIT-24 are used. Since the PIT does not need to be available for all the levels, it is memory efficient and can updated incrementally. The second one is *Lookup Table* (LT). We use LT-n to represent the lookup table of the size 2^n, which corresponds to the Leaves_Subtree(s, l, n) spanning from node s at level l in the Trie. For example, LT-8 represents a lookup table that contains 2^8 entries, which are the Leaves_Subtree(s, l, 8) spanning from an entry corresponding to node s in the PIT-l. The LTs are used to locate the BMP with the prefix of the DA as an index.

Format of Prefix Information Table

An Index to Output Port

Entry number	P	I	Output Port
	1-bit	1-bit	14-bit

An Index into Lookup Table

Entry number	P	I	Index into Lookup Table
	1-bit	1-bit	14-bit

Figure 3: PIT format.

As discussed earlier, we divide the Trie into three parts: Levels $0 \rightarrow 16$, levels $17 \rightarrow 24$, and levels $25 \rightarrow 32$, respectively. Two PITs, one at level 16 (PIT-16) and the other at level 24 (PIT-24), are also constructed to aid the lookup process. The PIT-n is constructed by taking the first n bits from all the prefixes in the routing table. Prefixes with the same first n bits will result in one entry in PIT$-n$. A PIT entry consists of the Entry Number (the first n bits of the prefix itself), a one-bit direction indicator I, a one-bit prefix indicator P, and a 14-bit pointer Ptr. The format of the entry is depicted in Fig. 3. There are four types of entries, as identified by the combination of the values of P and I:

- **Pure Prefix**: The entry itself is a prefix and there are no other nodes in the Trie that belong to the subtree expanding from the entry itself as a root. In this case, P=1, I=0, and Ptr has the value of the output port that the packet should be sent to get to its next hop.

- **Pure Marker**: The entry is created by one or more nodes in the Trie that belong to the subtree expanding from the entry itself as a root. The entry itself is not a prefix. For the PIT$-n$, the value of the entry is obtained by taking the first n bits from the prefixes. In this case, P=0, I=1, and Ptr points to the corresponding Lookup Table (discussed below).

- **Prefix and Marker**: The entry itself is a prefix and there are other nodes in the Trie that belong to the subtree expanding from the entry itself as a root. For the PIT$-n$, the value of the entry is obtained by taking the first n bits from the prefixes. In this case, P=1, I=1, and Ptr points to the corresponding Lookup Tables.

- **Non-Valid Entry**: The entry has no valid value. All the entries in PIT are initialized with P=0 and

I=0 until their values are assigned. This is used to aid the search of the BMP, as will be clear shortly.

Format of the Lookup Table

Lookup Tables (LTs) provide the mapping between the subtree and the output ports. There are three types of LTs with the proposed scheme:

- **LT-7**: It is an LT pointed to by an entry corresponding to node s in PIT-16. The entries of the LT consist of the leaves at the fully expanded Subtree(s, 16, 7) and provides a mapping between the leaf nodes and their corresponding output ports. It has 128 (2^7) entries.
- **LT-8**: It is an LT pointed to by an entry corresponding to node s in PIT-24. The entries of the LT consist of the leaves at the fully expanded Subtree(s, 24, 8) and provides a mapping between the leaf nodes and their corresponding output ports. It has 256 (2^8) entries.
- **LT-15**: LT-15 contains all possible mapping of the entries from level 0 to 15. As a trade-off between the lookup speed and memory usage, making the complete tree available reduces the lookup time with a modest increase of memory usage.

The mapping between the complete binary tree and the LT-8 can be seen in Fig. 4.

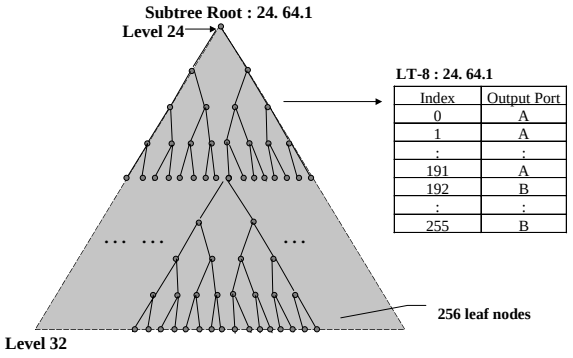

Figure 4: The structure of LT-8.

Note that it is relatively easy to construct the PITs and LTs from the routing table. As a trade-off between efficient memory usage and lookup throughput, PIT-16 is constructed as a fully-expanded table (2^{16} entries), while the PIT-24 is a hash table as full expansion requires a large amount of memory storage space.

2.2 The proposed IP lookup scheme for IPv4

The proposed IP lookup scheme for IPv4 can be summarized as follows:

When a packet arrives, the DA (bits $0 \rightarrow 31$) of the packet is extracted and the following procedure is followed.

With the notation DA_b^a being used to indicate the a-th to b-th bits of the DA, the aforementioned lookup scheme can be summarized as follows. (Note that the MSB of the DA is bit 0.)

Algorithm 1 *Prefix_Binary_Tree IP Lookup*

Input: DA,PIT-16,PIT-24,LT-7,LT-8,LT-15
Output: Output Port(OP)

$M_{16} \leftarrow$ PIT-16[DA_{15}^0]
if (M_{16}.P == 1 and M_{16}.I == 0)
    BMP_OP $\leftarrow$ M_{16}.Ptr
    Exit
elseif (M_{16}.I == 1)
    $M_{24} \leftarrow$ Hash_Search(DA_{23}^0, PIT-24)
    if (M_{24} is not nil and M_{24}.P == 1 and M_{24}.I == 0)
        BMP_OP $\leftarrow$ M_{24}.Ptr
        Exit
    elseif (M_{24} is not nil and M_{24}.I == 1)
        BMP_OP $\leftarrow$ LT-8[M_{24}.Ptr][DA_{31}^{24}]
        Exit
    elseif (M_{24} is nil)
        BMP_OP $\leftarrow$ LT-7[M_{16}.Ptr][DA_{22}^{16}]
        Exit
    end
else {M_{16}.P == 0 and M_{16}.I == 0}
    BMP_OP $\leftarrow$ LT-15[DA_{14}^0]
end

end

The lookup scheme is illustrated in Fig. 5.

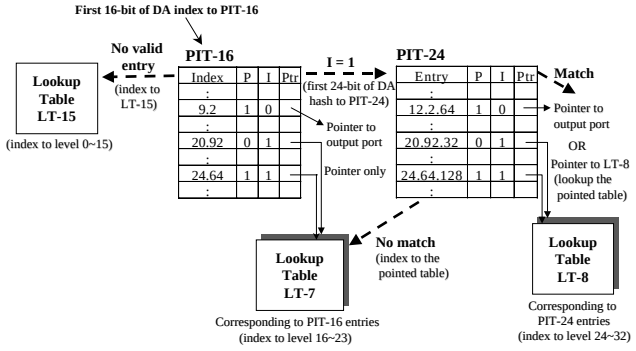

Figure 5: Flow chart for IP lookup.

As can be seen from the above description, the best case for the lookup needs one memory access (to PIT-16) and the worst case needs two memory accesses (to

PIT-16 and LT-8, respectively) and one hash (to PIT-24). More memory (about 300 Kbytes more) can be used to reduce the worst case access time to 2 memory accesses. This can be done by eliminating the PIT-24 and therefore the hash access can be avoided.

Figure 6 shows a possible hardware implementation of the proposed IP lookup scheme. Note that the dotted lines in the figure indicates that a pointer is pointing to a Lookup Table.

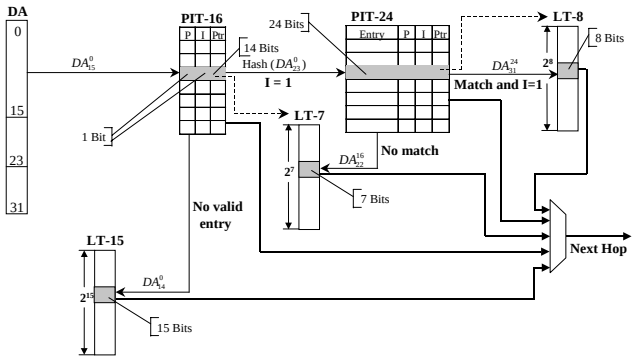

Figure 6: Hardware implementation.

2.3 Further improvements

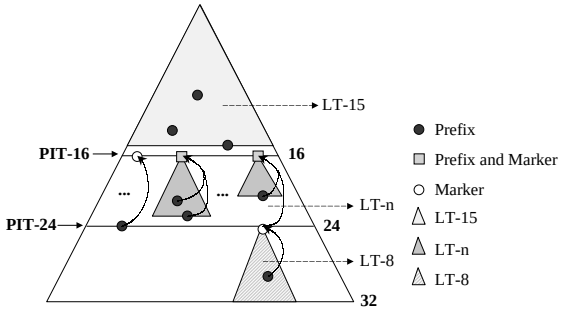

Figure 7: LTs of different lengths.

The basic scheme just described expands the PIT-16 to be a full table containing 2^{16} entries so as to reduce the memory access time. This is a trade-off between the needed memory accesses and the storage space requirements.

By observing that construct of LTs, we come up with some memory-conserving methods to save the needed storage space. In the aforementioned basic scheme, there are three kinds of LTs: LT-7, LT-8, and LT-15, respectively, in the forwarding tables. Note the LT$-n$ is a full table containing 2^n entries corresponding to the Leaves_Subtree(s, l, n) spanning from node s at level l in the Trie. However, since not all the LTs contain routing prefix entries that need to span down to the leaf nodes, there's no need to store the full tables for them. This is obvious from Fig. 7. By varying the depth of the subtrees, we can reduce the memory size needed by the LTs. However, when accessing the LTs, we need to know the corresponding LT size to know how many bits in the DA should be used in the index into the LT in question. Therefore, there should be an additional field in the entries of a PIT to store this information. Note that we do not change the LT-15. Therefore, three additional bits are needed so as to represent the length of the possible lengths of LTs. The format of the entry in a PIT for LTs with variable lengths is shown in Fig. 8.

An Index to Output Port

Entry number	P	I	LT_Depth	Index into Lookup Table
	1-bit	1-bit	3-bit	19-bit

Figure 8: PIT entry format for LTs with varying lengths.

Updates to the routing tables will result in changes to the forwarding tables (both PITs and LTs). However, with our forwarding table update mechanism the updates can be done easy and therefore limits the updates to a smaller amount of tables. Due to space limitations, the description of the update scheme is omitted here.

3 Performance evaluation

In proposing the IP lookup scheme, we have two targets in mind, one is to reduce the memory space needed for the forwarding tables, and the other to speed up the IP Lookup process. In this section, we show by simulation that the proposed scheme has a low memory requirement while maintaining a very high-speed IP lookup. Note that the current backbone routers have a routing table with about 45,000 entries [11]. We use the publicly available routing tables logs as a basis for comparison. These tables are offered by the IPMA project [12] and provide a daily snap shot of the routing tables used by some major Network Access Points (NAPs).

3.1 Memory requirements for the forwarding tables

The forwarding tables of the proposed scheme consists of two kinds of tables:

1. Prefix Information Tables:
   (a) PIT-16:
   - If Fixed-Depth LTs are used: The PIT-16 needs $2^{16} \times 2$ bytes.
   - If LTs with Varying-Depth are used: the PIT-16 needs $2^{16} \times 3$ bytes.

   (b) PIT-24: The PIT-24 needs (# of entries in PIT-24) $\times 5$ bytes.

2. Lookup Tables:

   (a) LT-15: The LT-15 needs $2^{15} \times 1$ bytes.

   (b) If Fixed-Depth LTs (LT-7 and LT-8) are used: The LTs need:

   $$(\#\text{ of LT-7s}) \times 2^7 + (\#\text{ of LT-8s}) \times 2^8 \text{ bytes}$$

   (c) If LTs with Varying-Depth are used: The LTs need:

   $$\sum_{n=1}^{8}((\#\text{ of LT-}n\text{'s}) \times 2^n) \text{ bytes}$$

The memory needed by the forwarding tables is the sum of the memory space needed by all the PITs and LTs. Note that the total memory requirement depends greatly on the number of entries in the tables. We use the routing table for Mae-East NAP, Mae-West NAP, PacBell NAP and AADS NAP [12] (considered the largest and can be representative of large back bone routers) as the bases to get the information about possible number of entries in different tables and then calculate the memory requirements of the proposed scheme.

Table 1 shows the memory requirement for the basic scheme (using Fixed-Depth LTs) under the four different NAPs. As can be seen, when used in the backbone with more than 45,000 routing entries, the total memory space needed by the basic scheme is about 670 Kbytes. If Varying-Depth LTs are used, the memory needed can be further reduced to about 560 Kbytes.

3.2 Lookup performance for IPv4

As discussed above, with the proposed IP lookup scheme, the worse case for a lookup for IPv4 will need two memory accesses plus one hash access. In lowing, we perform simulation runs of 200,00(entries against our proposed scheme which ta routing entries of backbone routers in Mae-Ea and Mae-West NAP in order to get average case mance data for the proposed scheme.

Table 2 shows the number of memory accesses for the basic scheme for the aforementioned sin scenario on different dates. Note that at the l umn, the Average Lookup Steps consists of "# (ory access" and "# of hashes". For example, erage lookup steps needed for Mae-East NAP (row), should be read as 1.39884 memory acces 0.54379 hashes. Under the heavy backbone tra with about 45,000 entries in the routing table, posed scheme needs in average 1.4 memory acce 0.5 hash.

4 An IPlookup scheme for IPv6

In this section, we show how the proposed described in the previous section can be scaled f

4.1 The scheme

The proposed IP lookup scheme for IPv4 can be extended form the basic lookup scheme described in Section 2 and is summarized as follows.

The addressing architecture of the IPv6 [13, 14, 15] provides a hint that by separating the multicast and IPv6 Aggregatable Global Unicast addresses, we can save spaces needed for the lookup tables. Note for the lookup for Aggregatable Global Unicast addresses, the Trie has been divided into 6 sections, with PITs available at levels 16, 32, 40, 48, 64, and 128. (This is to match the prefix length distribution of the 6Bone as shown in Table 3. and is also due to the fact that the bit length of the prefix of an IPv6 Aggregatable Global Unicast Address entry will not fall in between 64 and 127). In addition, we do not need to take care of the Link-Local or Site-Local addresses as they would not be processed by a router.) An entry in the PITs either points to LT-n or has the corresponding output port number. The procedure to construct the lookup tables (PITs and LTs) is very similar to that of the scheme for IPv4 and is omitted here. Based on the observation, the procedure described below is followed.

IP lookup mechanism for IPv6

When a packet arrives, the DA of the packet is extracted. If the first 8 bits of the DA is 0xFF, the packet is a multicast packet. Send the packet to the lookup logic that does a hash into PIT-128. (This is because that the number of multicast packets is relatively small when compared to Unicast packets.) If the first 3 bits is 001, the following procedure is followed. (The scheme is depicted in Fig. 9.)

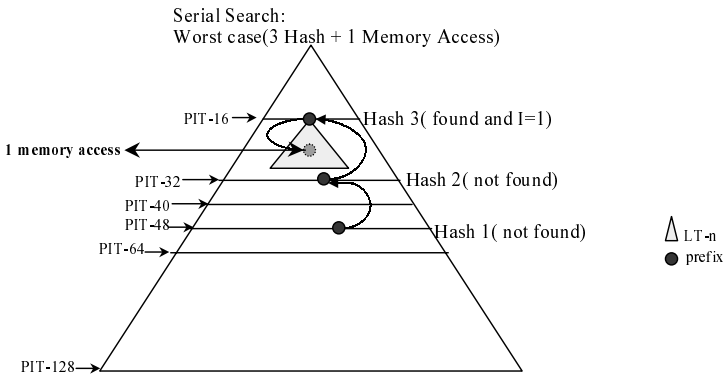

Figure 9: Extending the basic scheme for IPv6 lookup.

Take the first 48 bits of the DA as an index into the PIT-48.

1. If the entry has the values of P=1 and I=0, then the BMP has been found. The value of the Ptr is the output port. Exit.

Table 1: The memory requirements for the basic lookup scheme for IPv4.

Site	Date	Routing Entries	Number of LTs	Memory Usage with Fixed Depth (MB)	Memory Usage with Varying Depths (MB)
Mae-East NAP	6/17/99	45730	3094	0.6602211	0.55020142
Mae-West NAP	6/17/99	26747	2548	0.53955173	0.44644451
PacBell NAP	6/17/99	24555	2434	0.52044868	0.42983055
AADS NAP	6/17/99	14970	1831	0.41980839	0.3600893

Table 2: Lookup performance of the basic scheme.

Site	Date	Routing Entries	Number of LTs	Memory Usage with Varying Depths (MB)	Average Lookup Steps
Mae-East NAP	6/17/99	45730	3094	0.55020142	1.39884 + 0.54379 hash
Mae-East NAP	11/9/98	42741	2792	0.52557945	1.34254 + 0.54132 hash
Mae-West NAP	6/17/99	26747	2548	0.44644451	1.53533 + 0.53311 hash
Mae-West NAP	11/9/98	22152	2150	0.4100647	1.46475 + 0.50481 hash
Average					1.43537 + 0.53076 hash

Table 3: Prefix distribution of 6Bone(May 11, 1999).

	Prefix #	%
24 bits	54	24.2
25-47 bits	34	15.2
48 bits	118	53.0
49-63 bits	15	6.7
64 bits	2	0.9
Total	223	100

2. If the entry has the value I=1, this means that it is possible that there exists an entry with a longer prefix that matches the DA. In this case, extract the first 64 bits of the DA as a prefix and hash into PIT-64.
   (a) If there is a matching entry and the values of P=1 and I=0, the BMP has been found. The value of the Ptr is the output port. Exit.
   (b) If there is a matching entry and I=1, this means that it is possible that there exists an entry with a longer prefix that matches the DA. In this case, extract the first 128 bits of the DA as a prefix and hash into PIT-128. The entry that matches the hash provides the corresponding output port. Exit.
3. If there is no matching entry in PIT-48, this means that the BMP is a prefix of a length less than 48. In this case, extract the first 32 bits of the DA as a prefix and hash into PIT-32.
   (a) If there is a matching entry and the values of P=1 and I=0, the BMP has been found. The value of the Ptr is the output port.
   (b) If there is a matching entry and I=1, this means that the BMP is a prefix of a length between 32 and 47. In this case, extract the first 40 bits of the DA as a prefix and hash into PIT-40.
      i. If there is a matching entry and the values of P=1 and I=0, the BMP has been found. The value of the Ptr is the output port.
      ii. If there is a matching entry and I=1, extract bits $40 \rightarrow (39+n)$ as an index into the LT-n ($1 \leq n \leq 7$). Output the value of the output port. Exit.

iii. If there is no matching entry in PIT-40, this means that the BMP is a prefix of a length between 32 and 39. Take the Ptr obtained at Step 3b. The Ptr should point to a LT-n. Use bits $32 \rightarrow (31+n)$ as a index into the LT-n and get the corresponding value of the output port, which is the output port of the BMP ($1 \leq n \leq 7$). Exit.

(c) If there is no matching entry in PIT-32, this means that the BMP is a prefix of a length less than 32. In this case, extract the first 16 bits of the DA as a prefix and hash into PIT-16.

i. If there is a matching entry and P=1 and I=0, the BMP has been found. The value of the Ptr is the output port.

ii. If there is a matching entry and I=1, extract bits $16 \rightarrow (15+n)$ as an index into the LT-n ($1 \leq n \leq 15$). Output the value of the output port. Exit.

iii. If there is no matching entry in PIT-16, then the BMP is a prefix of a length between 4 and 15 (as the first 3 bits are fixed to be 001). Take the bits 4 to 15 of the DA as an index into LT-13 and retrieve the value of the output port. Exit.

Fig. 9 also shows the worst case scenario when 3 hashes and 1 memory accesses are needed to locate the BMP of an IPv6 DA. Note that counting the comparison of the first 8 bits of the DA to decide how to perform the lookup, the total time needed in the worst case in 3 hashes and 2 memory accesses.

4.2 Hardware implementation of the scheme for IPv6

We have implemented the proposed lookup scheme for IPv6. To the best of our knowledge, this is the first hardware implementation that provides solutions to the IP lookup for IPv6. The FPGA implementation is named IPLU0628 and it currently requires an external hash device to provide the hash function and an external two-way SRAM to save the PITs and LTs. An FPGA-proven synthesizable Verilog RTL code is available.

4.2.1 functional block diagram and Pin description

Fig. 10 shows the functional block diagram of the IPLU0628. The description of the pins is as follows.

CLK (clock) This is a free running clock used to control all operations inside the IPLT0628. The rising edge of this clock is the timing reference.

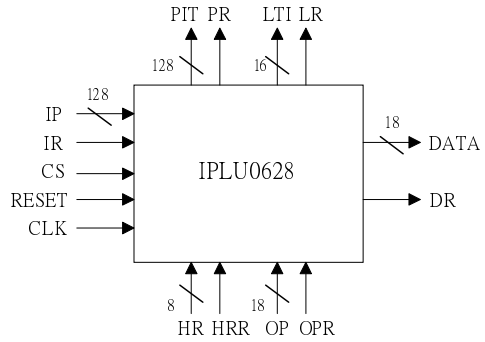

Figure 10: Functional block diagram of the FPGA implementation.

RESET (Reset) This is a synchronous, active-high control input providing the hardware reset for the device. During the initialization, the signal must be assert high for at least one clock cycle. This will set all output into high-Z.

CS (Chip Select) This is an asynchronous, active-high control input. When assert high, the IPLU0628 is in normal function mode. When assert low, the device is in standby mode.

IR (IP Ready) This is a synchronous, active-high control input. It indicates whether the IP address is ready for next lookup operation. Whenever the IPLU0628 completes the current lookup operation and finds IR high, the next lookup operation is launched.

IP (Ipv6 Address) This data input bus carries 128-bit IPv6 address into the IPLU0628.

PR (PIT Ready) This is a synchronous, active-high control output. It indicates whether the PIT hash index is ready. Whenever the hash device finds rising edge, a new PIT hash index is ready.

PIT (PIT Hash Index) This data output bus carries 128-bit PIT hash index into the hash device.

MC (MultiCast) This is a active-high control output. It indicates whether the PIT hash index is multicast or unicast.

HRR (Hash Result Ready) This is a synchronous, active-high control input. It indicates whether the PIT hit is ready. When IPLU0628 finds the rising edge, a new HR is ready.

HR (Hash Result) This data input bus carries 8-bit PIT hit data into the IPLU0628.

LR (LTI Ready) This is a synchronous, active-high control output. It indicates whether the LTI is ready. Whenever the hash device finds the rising edge, a new LTI is ready.

LTI (Lookup Table Index) This data output bus carries 16-bit lookup table index into the hash device.

OPR (Output Port Ready) This is a synchronous, active-high control input. It indicates whether the output port is ready. Whenever for IPLU0628 finds the rising edge, a new output port is ready.

OP (Output Port) This data input bus carries 18-bit output port data into the IPLU0628

DR (Data Ready) This is a synchronous, active-high control output. It indicates whether the DATA is ready.

DATA (output port) This is an 18-bit output bus. It sends out the output port data for using the IP lookup for current IP.

4.2.2 FSM description and simulation

We use the Xilinx State Editor to design a state machine description, synthesis HDL code, check syntax, and create the corresponding macro. A waveform diagram is presented in Fig 11. It is simulated by using a script macro. In this example, an IP address was accepted and a PIT hash index is received as a result. Depending on the result received and the Lookup Table Index (LTI) the output port is then found.

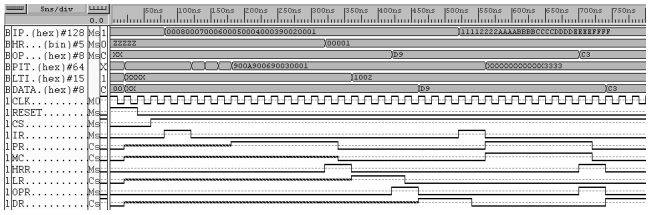

Figure 11: The waveform of the simulation result.

4.2.3 reference system configuration

The FPGA implementation can be deployed for IP lookup using the reference system configuration described in Fig. 12.

4.2.4 Exploring the parallelism

The lookup performance for IPv6 can be further improved by exploring the parallelism of the basic lookup scheme. The parallel architecture is depicted in Fig. 13.

5 Conclusion

One of the major bottlenecks in the Internet router design is the slow, software-based IP lookup procedures. In this paper, we have propose a high-speed IP lookup scheme for IPv4 by using *forwarding tables* consisting of Prefix Information Tables (PITs) and Lookup Tables (LTs) that provide guidelines for efficient search. The basic scheme needs 1 memory access in the best case, and 2 memory accesses plus one hash in the worst case to locate the BMP. The scheme requires only 560 KBytes to 670 KBytes of memory space when routing tables with about 45,000 entries (representative of the entries in backbone routers) are used for simulation. When 50ns DRAM is employed to store the forwarding tables, the lookup scheme for IPv4 offers lookup speed of 10 millions packets per second (10 MPPS). The speed can be improved linearly with the speedup of the type of memory used. The operation of the scheme is simple, and a possible hardware implementation is provided. The effects of updates to the routing table is found to be minimum. Compared to the related work in the literature, the scheme provides an efficient use of memory and at the same time, maintains the high-speed lookup required by the next generation IP routers.

The scheme is then shown to be scalable for IP lookup in IPv6. A detailed description of the extended scheme and its FPGA implementation are provided to show its feasibility. Although the routing entries and traffic traces for IPv6 do not have the scale comparable to their readily available counterparts in IPv4, initial analysis shows that the scheme can be quite effective as it requires in the worst case 3 hashes and 2 memory access for the lookup in IPv6. Further improvement to the lookup performance can be achieved by exploring the parallelism of the basic lookup mechanism. In this case, only 1 hash and 2 memory access are needed.

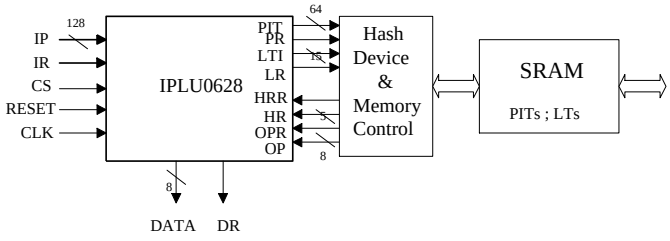

Figure 12: The reference system configuration.

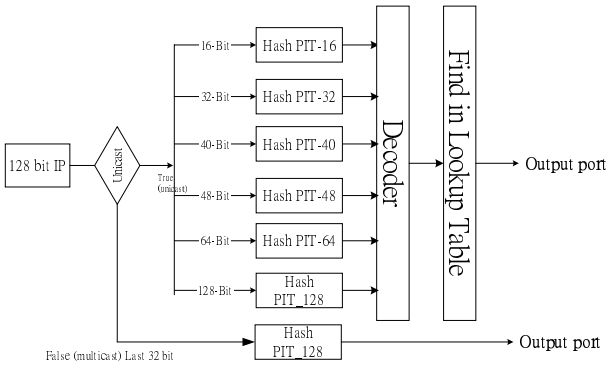

Figure 13: Parallel architecture for IPv6 lookup mechanism.

(Note that the access time can be reduced with more memory used.)

The memory requirement for the proposed IPv6 lookup scheme can only be analyzed using hypothetical routing entries. We will investigate how more real-life IPv6 traffic information can be gathered to further validate the schemes. We are currently working on the issues.

Acknowledgement

The authors would like to thank Prof. W. K. Lai and his research group at National Sun Yat-sen University for providing us with the IPv6 traffic information and valuable comments/dicssusions.

References

[1] S. Deering and R. Hinden, "Internet Protocol version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[2] "High-Speed Routing Table Search Algorithms," Torrent Networking Technology Technical Paper, See http://www.torrentnet.com/.

[3] M. Degermark, A. Brodnik, S.Carlsson, and S. Pink, "Small Forwarding Tables for Fast Routing Lookups," in *Proc. ACM SIGCOMM '97*, Cannes, France, Sept. 1997, pp. 3–14.

[4] W. Doeringer, G. Karjoth, and M. Nassehi, "Routing on Longest-Matching Prefixes," *IEEE/ACM Trans. on Networking*, vol. 4, no. 1, pp. 86–97, Feb. 1996.

[5] P. Gupta, S. Lin, and N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds," in *Proc. IEEE INFOCOM '98*, San Francisco, USA, March 1998.

[6] N.-F. Huang, S.-M. Zhao, and J.-Y. Pan, "A Fast IP Routing Lookup Scheme for Gigabit Switch Routers," in *Proc. IEEE INFOCOM '99*, New York, USA, March 1999.

[7] C. Labovitzs, G. R. Malan, and F. Jahanian, "Origins of Internet Routing Instability," in *Proc. IEEE INFOCOM '99*, New York, USA, March 1999.

[8] S. Nilsson and G. Karlsson, "Fast Address Lookup for Internet Routers," in *Proc. IFIP 4th International Conference on Broadband Communications (BC'98)*, 1998, pp. 11–22.

[9] S. Nilsson and G. Karlsson, "IP-Address Lookup Using LC-Tries," *Submitted to IEEE Journal on Selected Areas in Communications.*

[10] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, "Scalable High Speed IP Routing Lookups," in *Proc. ACM SIGCOMM '97*, Cannes, France, Sept. 1997, pp. 25–36.

[11] S. Bradner, "Next Generation Routers Overview," in *Proc. Networld Interop 97*, 1997.

[12] Merit Networks, Inc, "Internet Performance Measurement and Analysis (IPMA) Statistics and Daily Reports," See http://www.merit.edu/ipma/routing_table/.

[13] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 2373, July 1998.

[14] R. Hinden, M. O'Dell, and S. Deering, "An IPv6 Aggregatable Global Unicast Address Format," RFC 2374, July 1998.

[15] R. Hinden and S. Deering, "IPv6 Multicast Address Assignments," RFC 2375, July 1998.