International Journal of Computer Networks and Applications
Volume 3, Issue 1, January – February (2016)

RESEARCH ARTICLE

IP Address Lookup in an IP Router Based On a Reorganized Binary Prefixes Value Tree (RBPVT)

Haouassi Hichem
Univ Khenchela, Fac. ST, ICOSI Lab., BP 1252 El Houria, 40004 Khenchela, Algeria.
Houassi_h@yahoo.fr

Maarouk Toufik Mesaaoud
Univ Khenchela, Fac. ST, ICOSI Lab., BP 1252 El Houria, 40004 Khenchela, Algeria.
toomaarouk@yahoo.fr

Mahdaoui Rafik
Univ Khenchela, Fac. ST, ICOSI Lab., BP 1252 El Houria, 40004 Khenchela, Algeria.
Mehdaoui.rafik@yahoo.fr

Abstract – IP address lookup to route data packets is an important function in a router and improving this function improves the overall performance of the router. From the data structures used for the prefixes representation, there are the trees that represent prefixes with their binary values. However, this data structure requires an improvement because of the longest prefix function search complexity. Our approach is used to improve the routing information search time in the prefixes values binary tree by periodically reorganizing the tree according to the use of prefixes; the most recently used prefixes are stored in the higher levels of the reorganized binary prefixes value tree (RBPVT) which improves the data packets routing time. The tests and evaluation of the access memory number of the longest prefix match search algorithm shows that the IP address lookup algorithm based on our RBPVT tree improves the performance of the IP routers in terms of average memory access number.

Index Terms – IP routers, IP address lookup algorithm, CIDR, binary prefixes value tree.

1. INTRODUCTION

On the Internet, the communication between machines is performed by packets of information [1] [2][3]. Once the machines transmit their packets in the network, there are the routers that forward the packets on network links to their final destinations. The rapid growth in the number of Internet users results the increase in the size of routing tables and complicate the data packets routing operation.

To forward packets, the routers must determine where to send each incoming packet. More exactly, the routers must be finds for each incoming packet, the next-hop address of the router and their output port number through which the packet should be sent. We call these concepts the "routing information".

The router use the destination IP address of each incoming packet and searches the routing information in its forwarding table for decide where to send it next, this operation is called IP address lookup. More specifically, the router searches in their forwarding table to find the longest prefix matching the destination IP address of the incoming packet. To do this, the prefixes are compared bit by bit to the IP address of incoming packets and that the routing information associated with the longest of the matching prefixes should be used to forward the packet.

The entries in the forwarding tables have been increasing as the number of Internet users is increasing, the longest prefix matching search operation becomes very complicated, and for this, several researches and studies have been proposed in the literature that have addressed this problem and improve the router's performance which is directly related to packet processing time.

In this paper, we propose a data structure for presenting the prefixes of a routing table in order to improve the time search of the longest prefix matching IP address in routers. Our data structure is called reorganized binary prefixes value tree (RBPVT) and it is based on the principle of reorganization of the binary prefixes value tree proposed in the literature [4], our tree is restructured periodically to store the most recently used prefixes in the nodes of higher levels of the tree to accelerate the prefixes research operation.

The rest of the paper is organized as follows. Section 2 reviews the previous works. In section3, we present our detailed data structure and IP address lookup algorithm, and finally, Section 4 presents the performance evaluations and comparisons with other previous works followed by conclusion and future works.

RESEARCH ARTICLE

2. RELATED WORK

In the following sections we introduce and classify some software approaches that are proposed to tackle the IP address lookup operation problem.

The majority of all recent algorithms use trees data structures to represent the prefixes stored in the routing table [5], [6], [7], we have classified the existing trees data structure according to the principle of prefixes representation in the trees into binary representation of the prefix, transformation and representation prefixes in the form of IP address ranges, and binary prefixes values tree.

2.1. Binary trees prefixes

The prefixes are bit strings of varying lengths [8]; they can be naturally represented by a binary tree [11].

Binary tree is a simple data structure which represents prefixes with paths from the root to the leaf; the next hops are stored in the internal nodes or the leaves, however the number of memory accesses required evaluating the next hop is much higher. In order to reduce the depth of the tree, Multibit trees [9], [10], [11] are one way to reduce the memory accesses number and to optimize the time needed to find the longest prefix.

In the inner of the binary tree may exist internal nodes with only one-child nodes, these nodes must be traveled even though no branching decision is made which remove the search speed. A levels compression and paths compression techniques are used to remove this problem, such as LC-trie [12], [13] and patricia trie [14].

On the other hand, most tree data structures include many empty internal nodes i.e. to not store any information. The authors of the paper [15] propose a longest prefix search algorithm based on a data structure called priority trie. This algorithm exploits the empty internal nodes of the binary tree to store the longest prefix among the prefixes belonging to a sub-tree rooted by this empty node. Hence the memory space occupied by the tree is reduced. In [16], the authors propose a longest prefix matching search algorithm using a binary tree with dynamic content, the content of this tree varies according to the IP addresses handled by the router. This algorithm exploits the empty internal nodes of the binary tree by placing prefixes copies of the least recently used nodes in the higher levels empty node of the tree. The binary trees prefixes does not depend on the length of prefixes, it can be readily migrated to IPv6.

2.2. Prefix Range Search

As prefixes are of arbitrary lengths and can be represented by intervals (the start point and end point), some algorithms have been proposed to search with these endpoints. The authors of [17] use a binary search on the end points representing the prefixes, but pre-computation is necessary for the data structure updates operations. In [18], an algorithm is proposed to improve the performance by using a B-tree structure to prevent pretreatment of the data structure in [17], but each prefix may be stored in multiple nodes. In [19], a new algorithm is proposed to improve performance by storing one node for each prefix. In [20] another tree data structure is used to store the end points of the ranges of IP addresses (*height-balanced inorder-threaded binary search tree*).

All these range-based algorithms can achieve the worst case lookup performance of O(logN) (N is the number of prefixes in a routing table) [21]. These algorithms are expected to be good, with respect to the scalability to a large routing table or to the migration to IPv6, since are performs the balanced binary search and does not depends on the length of prefixes [5].

2.3. Binary search based on prefix value

The binary search algorithms based on prefixes values are used to provide trees without empty nodes i.e., all nodes in the tree stores data. In order to search on the binary values of the prefixes, the latter should be sorted according to their binary values. The mechanisms based on the binary prefixes values provide a technique for comparison of different prefix lengths [22]. To reduce the depth of the tree, the weighted prefix tree (WPT) [23] considers the number of descendents in the choice of the root of each level. The WPT tree constructed has a shorter depth and is more balanced than the binary tree prefixes. The multiple balanced prefix trees (MBPT) [24] constructs more balanced trees with disjoint prefixes only [25]. The longest prefix first search (LPFST) [26] place the longest prefixes in the upper levels of the tree such that the length of the prefix stored in each node is greater than or equal to the lengths of prefixes of his child, by this technique the routing information search process ends when the destination address of the incoming packet matches a prefix before arriving at a leaf. The authors of [26] propose a balanced binary search algorithm based on the binary search tree. This algorithm builds a binary search tree with only the leaves of the classical binary tree and create in every node a vector for contains a nested prefixes.

Most of these proposed techniques use balanced trees that uniform search time in the tree, but the problem is that the depth of these trees generally depends on the number of prefixes in the routing table, and then it is possible arriving at a complexity which exceeds O(w) where w is the maximum length of prefixes.

In this paper we proposed an IP address lookup algorithm that uses a Binary tree where the most recently used prefixes are firstly founds in the tree. In our proposed algorithm, the most recently used prefixes for previous forwarded packet are stored in the upper levels of the tree which speeds generally the IP address lookup operation in the forwarding table. Our tree can

RESEARCH ARTICLE

be classified in the last class of data structures presented above in the state of the art.

3. PROPOSED ALGORITHM

In the classical binary tree each prefix is represented by a path from the root to a node of the tree, but the tree contains empty nodes that do not store any information which consumes more memory space. Empty nodes increase the height of the tree, and consequently increase the search speed. To eliminate these problems the binary search algorithms on the prefix values are used to provide trees without empty nodes i.e. all tree nodes stores useful data. In this kind of tree, prefixes should be sorted according to their binary values [22].

The objective of our proposition is to accelerate the longest prefix search operation in the tree by storing the most recently used prefixes in high levels of the prefix value tree. To determine the most recently used prefixes, we propose a mechanism to evaluate the use of prefixes stored in the tree, our mechanism associated with each prefix a variable containing a value called "priority value" that is reduced (evaporated) with the passage of time and increased by a constant value when accessing the node that contains this prefix, This value is updated with the longest prefix match search operation. Our longest prefix search algorithm updates the priority value using the formula 1. Then we compare the prefixes according to this value, if this value is higher, the prefix is considered mostly used and must be placed in a high level in tree.

On the other hand we have proposed a periodic evaluation and reorganizing of the prefix tree in order to take account of the new priority values of prefixes. Our algorithm has to reconstruct the tree of prefixes again if the majority of prefixes are located in positions not meeting its priority values. To decide a reorganization of the tree, it must firstly, evaluate the organization of prefixes in tree using the formula 2. For reorganize the tree, firstly, sort the prefixes according to their priority values, and then insert the prefixes one by one in the tree according to the comparative relationship proposed in [22] and initialize their priority values.

3.1. Structure of a tree node

The figure 3 illustrates the structure of a node in the reorganized binary prefixes value tree proposed. The tree node stores the prefix, the output port (OP), the priority value (PV) of the prefix, the last priority value updates time (LPVUT), and two pointers to the left and right child of the node.

Prefix	OP	VP	LPVUT	Child_ L	Child_ R

Figure 1 Node structure

Prefix: The prefix of the routing table

OP: The output port matching the prefix in the routing table.

Child_L and Child_R: Pointers to the left and right childs of the node respectively, or Null.

PV: The priority value of the prefix.

LPVUT: The time of the last priority value updates.

3.2. Calculation of the prefix priority value

To search the output port to an IP address of an incoming packet, the algorithm traverses the tree that represents all prefixes in the routing table using the bits of the IP address to find the longest prefix in one of the tree node that matches the IP address.

In our proposition the longest prefix search algorithm has updates the priority value field associated with each node contains a prefix in the tree that allows us to take an idea on the use of prefixes in the routing table by the router.

The priority value associated with a prefix reduced (evaporated) with the passing of time and increased by a constant value when accessing the node containing the prefix in the tree. The priority value is updated according to the following formula:

$$Pv(t+\Delta t) = \rho \cdot \frac{1}{\Delta t} \cdot Pv(t) + Q$$

Where,

$\rho \in [0,1[$: Is the evaporation coefficient which defines the decrease rate (evaporation) of the priority value between time t and t+Δt.

Δt: Is the time interval between two successive prefix accesses.

$V_P(t)$: Is the priority value of the prefix P at time t.

Q: is a constant value added to the old priority value when accessing the prefix.

The choice of ρ is important, because if ρ gets too close to 1, then priority value stagnation is observed. Similarly, choosing $\rho \approx 0$ implies too rapid decrease (evaporation) of the priority value, then, leads to select the most used prefix but not the most recently used.

3.3. Tree organization quality

The tree organization quality depends on the prefixes positions in the different levels of the tree, the tree is good organized if the most recently used prefixes are in the upper levels of the tree and the least recently used prefixes in the lower levels. To better understand the evaluation technique of the tree organization quality we provide the following definitions:

Definition1: Prefixes priority value

RESEARCH ARTICLE

The prefix priority value *Pv* is a real value associated with the prefix in the tree, this value represents the density of the recent prefix uses by the router.

Definition2: level priority value

The level priority value of the prefixes tree *LPV(i)* is the mean of the priority values of the prefixes stored in the nodes of a level *i* in the tree.

Definition3: tree organization quality

The tree organization quality is calculated using the following formula:

$$TOQ = \frac{\sum_{i=1}^{N} (LPV(i+1) - LPV(i))}{N-1}$$

Where,

TQO: Is a value representing the tree organization quality.

LPV(i): Is the priority value of a level *i* in the prefixes tree.

N: Is the levels number of the prefixes tree (tree depth).

The prefix tree is of perfect quality if and only if, whatever two prefixes *Pi* and *Pj* stored in the nodes of two levels *i* and *j* such as *i* is less than *j* implies that the *Pi* priority value is greater than *Pj* priority value.

Really almost impossible to organize the prefixes in a perfect quality tree since during construction of the prefix tree must be taken into account two parameters: the priority value and the comparison between prefixes defined in [22].

3.4. Building the proposed reorganized tree

The proposed tree is initially constructed as a binary tree of prefixes proposed in [22], is a tree without empty nodes i.e., all nodes in the tree store useful data. To search on the binary values of the prefixes, the latter must be sorted according to their binary values.

In our prefix tree, prefixes priority values are modified over time, our algorithm periodically evaluate the tree organization quality based on formula 2, if the organizational quality value exceeds a threshold α, it causes the shaft rebuild operation as follows: firstly, sorted in descending order every tree prefixes according to its priority values, then insert the prefixes one by one in the new tree based on the comparison of relationship between the binary values of prefixes defined in [22].

```
/* O_Tree and N_Tree: The old and new tree respectively */
Procedure Construction_Tree(O_Tree,N_Tree)
  {  List← Sort_Tree(O_Tree)
  While (List<>Null)
   {Insert_element(N_Tree,
     Head_List(List))
    List← Next (List)}
  }
  /* the function Sort_Tree sort the elements of the old tree
in descending order according to priority value PV */
Procedure Insert_element(Tree, Prefix)
  {If (Tree = Null) {
    Create_Node(Tree)
    Tree↑.Prefix←Prefix;
    Tree↑.PV ← 0 /*PV is a
                  priority value */}
  Else
   If Prefix < Tree↑.Prefix
  Insert_element(Tree
  ↑.Child_L, Prefix)
   Else
    Insert_element(Tree ↑.Child_R, Prefix)}
```

Figure 2 shows the initial binary tree applied to the prefixes of table 1. The prefixes priority values are initialized to 0.

Prefix	Length	Output	Prefix	Length	Output port
00*	2	A	111111	6	F
010*	3	B	110100	6	G
101*	3	C	110101	6	H
1*	1	D	111100	6	I
111*	3	E	1100*	4	J

Table 1 Example of routing table

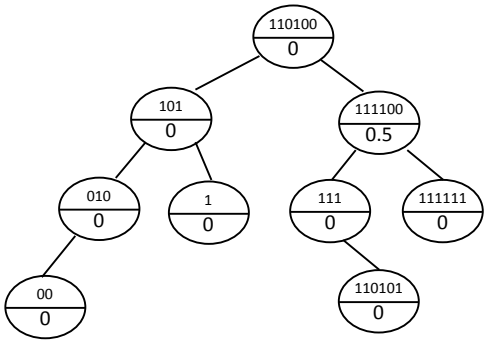

Figure 2 The RBPVT initial tree created from Table 1

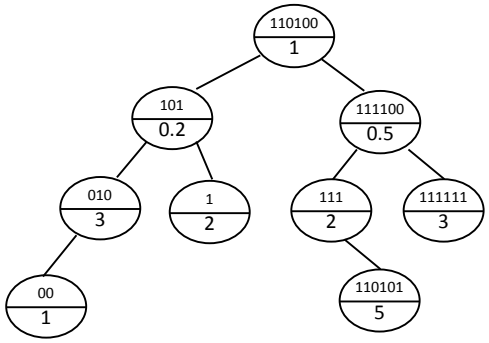

Figure 3 RBPVT tree with modified priority values

RESEARCH ARTICLE

It is assumed that after a period, prefixes priority values are modified as shown in the tree in Figure 3.

Assuming that the maximum of the organization quality threshold value of the tree $\alpha = 0.5$, the organization quality value is 0.66, this value is calculated by the formula 2 based on the priority values of the tree in figure 2, then the tree must be rebuilt again as shown in figure 4.

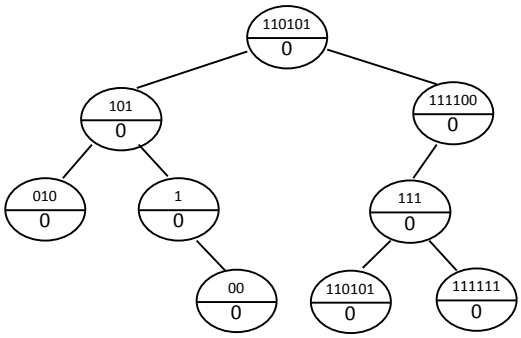

Figure 4 The RBPVT tree rebuilt from the tree of figure 3

We note in the tree of figure 4 that the prefixes have high priority values are mounted to higher levels in the tree.

3.5. The longest prefix search

The search procedure is simple, recursive and similar to the search procedure in a binary search tree. The procedure returns the longest prefix match if it exists and NULL otherwise.
It is important to note that the search procedure always starts with the most recently used prefixes that are stocked in the higher levels of the tree which improves the overall search time. The search procedure is as follows:

Tree: Is the RBPVT Tree.
ADR_IP: Is the IP address of the input packet.
PV: The priority value of the prefix.
LP: The longest prefix.
Q and ρ: Are constants.
Δt: Is the time interval between the time of the last update and the current time.

```
Procedure Search (Tree, Adr_IP)
 {If Tree = NULL then Return
     NULL
  Else
    If (Adr_IP match
      Tree↑.prefix)then
      LP ← Tree↑.prefix;
      Return LP ;
      Tree↑.PV← ρ *(1/Δt)*PV+Q
    Else
    If (Adr_IP<Tree↑.prefix)then
   LP←Search(Tree↑.Child_G,
     Adr_IP)
    Else
   LP←Recherche(Tree↑.
     Child_D, Adr_IP);  }
```

4. TESTS AND RESULTS ANALYSIS

During testing of our algorithm, we used 3 routing table of different sizes:

Routing tables	Sizes
Table 1	5000 prefixes
Table 2	10000 prefixes
Table 3	15000 prefixes

Table 2 Size of routing tables used during the tests.

We implemented the data structure and the proposed algorithm and we simulate the longest prefix search operation in an IP router. During the simulation we consider the following two criteria: average number of memory access and total number of memory access.

For each simulation scenario we test our algorithm with a manufacturing routing table and multiple data streams (set of data packets).

Scenario 1

In the first scenario we used the routing table1 contains 5000 prefixes and it tested for the search algorithm using three data streams in the form of packets tables. The sizes of the packets tables used are the following in table 3:

Paquets tables	Sizes
Table 1	1100 paquets
Table 2	1300 paquets
Table 3	1500 paquets

Table 3 Packets tables sizes used in the first scenario.

Scenario 2

In this scenario the second routing table of Table 2 is used (10000 prefixes) and the search algorithm is tested with three data stream, where, the sizes are as follows:

Paquets tables	Sizes
Table 1	3100 paquets
Table 2	3200 paquets
Table 3	3300 paquets

Table 4 Packet tables size used in the second scenario.

Scenario 3

In the third scenario it used the third routing table contains 15000 prefixes, so we tested our search algorithm on three other data stream, where, the sizes are as follows:

RESEARCH ARTICLE

Paquets tables	Sizes
Table 1	8000 paquets
Table 2	8050 paquets
Table 3	8100 paquets

Table 5 Packet tables sizes used in the third scenario.

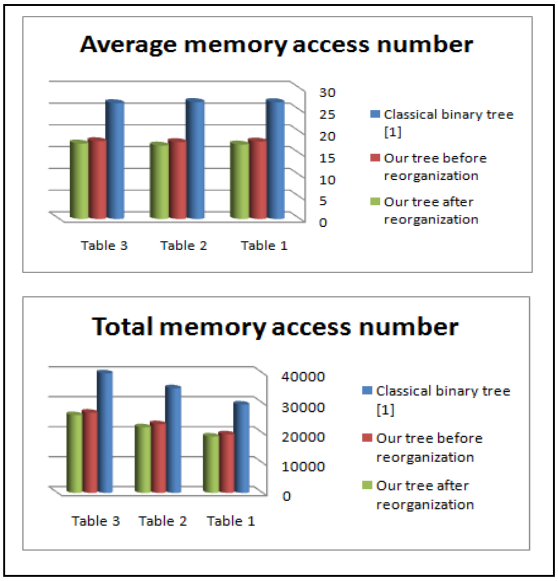

Figure 5 Results of the first scenario with a routing table 1.

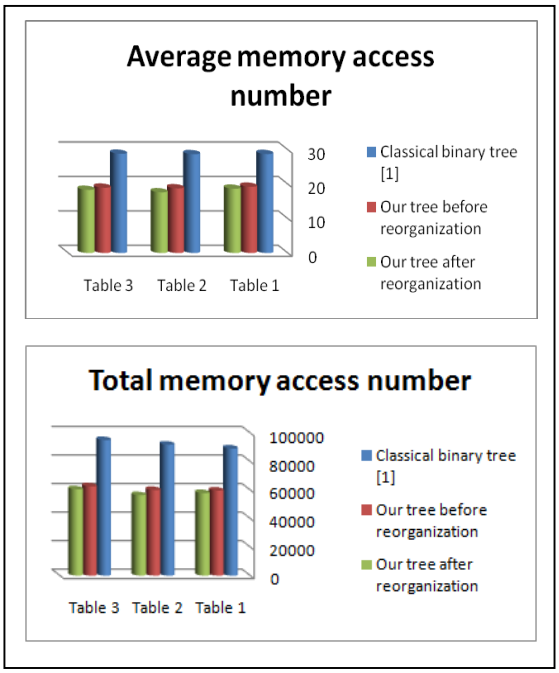

Figure 6 Results of the second scenario with a routing table 2.

The figures 5, 6 and 7 shows the comparison results of our data structure before and after reorganization with the classical binary tree data structure of prefixes [1] testes in the three previous scenarios. The comparison is in terms of average and total memory access number of the longest prefix search algorithm using several routing tables and several data streams (packets tables).

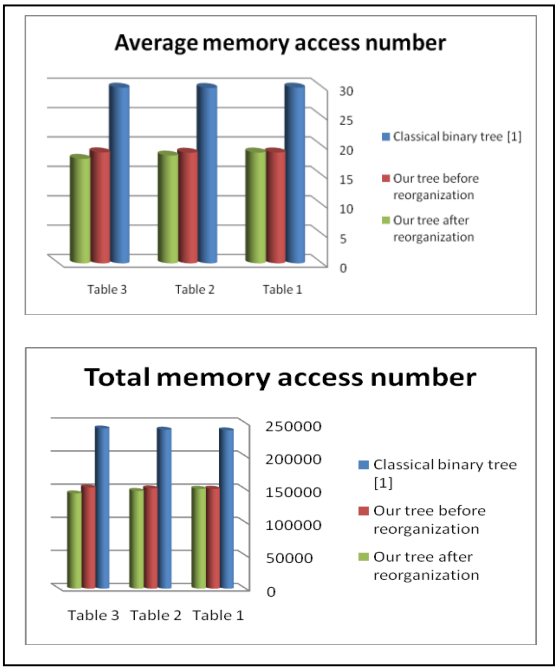

Figure 7 Results of the third scenario with a routing table 3.

In all scenarios we noticed that our tree presents considerable improvement of two tested criteria (average number of memory accesses and total number of memory access) before and after reorganization compared with the classical binary tree[1], on the other hand we found that our tree after reorganization has a small improvement in terms of average memory access number from the same tree before reorganization and since the most recently used prefixes are stored after reorganization of the tree in the nodes of the high levels of the tree which improves the prefix search time.

Once a prefix is used for forwarding the first packet of a data stream, it is sure that the router reuses the same prefix for routing packets of the same flow remaining, and then the prefix is found after reorganization in higher levels of the tree.

5. CONCLUSION

An IP address lookup algorithm in the IP routers based on reorganized binary tree has been proposed. The data structure used for representing the prefixes is in the form of a prefixes values binary tree, this tree is reorganized periodically to improve the longest prefix match search time, the tests and

RESEARCH ARTICLE

evaluation of our approach demonstrates that the search time of an IP address match search algorithm based on the prefixes values binary tree after reorganization is better than the same search algorithm using the tree before reorganization. Our proposed approach improves search time by storing the most recently used prefixes in higher levels of the tree after reorganization.

REFERENCES

[1] Y.P. Dalal, U.D. Jha, R.K., "Security Comparison of Wired and Wireless Network with Firewall and Virtual Private Network (VPN)," in Recent Trends in Information, Telecommunication and Computing (ITC), 2010.
[2] B. Ramakrishnan , S. R. Sreedivya , M. Selvi , "Adaptive Routing Protocol based on Cuckoo Search algorithm (ARP-CS) for secured Vehicular Ad hoc network (VANET)", International Journal of Computer Networks and Applications (IJCNA) Volume 2, Issue 4, pp 173—178, 2015.
[3] R. Vinuraj, SJ. Weta, "Application of Modified ACO Meta heuristic in Spray and Wait Routing", International Journal of Computer Networks and Applications (IJCNA) Volume 2, Issue 5, pp. 232—241. 2015.
[4] Antos, D.: Overview of data structures in IP lookups. CESNET Tchnical Report, (2002).
[5] Hyesook, L., and Nara L.: Survey and Proposal on Binary Search Algorithms for Longest Prefix Match, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, Vol. 14, No. 3, pp. 681—697, (2012).
[6] T. Yang, G. Xie, Y. Li, Q. Fu, A. X. Liu, Q. Li, and L. Mathy. Guarantee IP Lookup Performance with FIB Explosion. In ACM SIGCOMM, pp. 39–50, 2014.
[7] H. Lim, K. Lim, N. Lee, and K.-H. Park. On adding bloom filters to longest prefix matching algorithms. IEEE Transactions on Computers (TC), 63(2): pp.411–423, 2014.
[8] S. V. Limkar, R. K. Jha, and S. Pimpalkar, "Ipv6: issues and solution for next millennium of internet," in Proceedings of the International Conference & Workshop on Emerging Trends in Technology, ser. ICWET '11. New York, NY, USA: ACM, 2011, pp. 953–954.
[9] Wuu, L.C., Liu, T.J., Chen, K.M.: A longest prefix first search tree for IP lookup. Computer Networks, Vol. 51, Issue. 12, pp. 3354--3367 (2007).
[10] Moestedt, A., Sjodin, P.: IP address lookup in hardware for high speed routing. Hot Interconnects VI, (1998).
[11] Srinivasan, V., Varghese, G.: Fast address lookups using controlled prefix Expansion. Proceedings of ACM Sigmetrics, Vol. 17, Issue. 1, pp. 1--40. (1999).
[12] Nilsson, S., Karlsson, G.: IP-address lookup using LC-Tries. IEEE Journal on selected areas in communications, Vol.17, Issue. 6, pp. 1083--1092. (1999).
[13] Ravikumar, V.C., Mahapatra, R., Liu, J.C.: Modified LC-Trie based efficient routing lookup. Proceedings of the 10th IEEE MASCOTS 02, pp. 177 --182. (2002).
[14] Morrison, D.R.: PATRICIA - practical algorithm to retrieve information coded in alphanumeric. ACM, Vol. 15, No. 14, pp. 514--34. (1968).
[15] Lim, H., Mun, J.: An efficient IP address lookup algorithm using a priority-trie. IEEE, Global Telecommunications Conference, pp. 1--5. (2006).
[16] Houassi, H., Bilami, A.: IP address lookup algorithm using a dynamic content binary trie. IRECOS, Vol. 5, NO 3, pp. 337--341. (2010).
[17] Lampson, B. Srinivasan, V., Varghese, G.: IP lookups using multiway and multicolumn search. IEEE/ACM Networking, Transactions, Vol. 7, Issue. 3, pp. 1248--1256. (1999)
[18] Suri, S., Varghese, Warkhede, G. P.: Multiway range trees: Scalable IP lookup with fast updates. IEEE, GLOBECOM '01. Vol. 3, pp. 1610--1614. (2001).
[19] Lu, H., Sahni, S.: A B-Tree dynamic router-table design. IEEE Computers Transactions, Vol.54, Issue: 7, pp. 813--823. (2005).
[20] Li, Y.K., Pao, D.: Address lookup algorithms for IPv6., IEE Proceding on Communication, Vol. 153, Issue. 6, pp. 909 --918. (2006).
[21] Sun, Q., Zhao, X., Huang, X., Jiang, W., Ma1, Y.: Scalable exact matching in balance tree scheme for IPv6 lookup. IPv6'07, Kyoto, Japan, Copyright ACM 978-1-59593-713. (2007).
[22] Yazdani N., Min, P.S.: Fast and scalable schemes for the IP address lookup problem. Proceedings of the IEEE Conference on High Performance Switching and Routing, Vol.3, pp. 1610--1614. (2000).
[23] Yim, C., Lee, B., Lim, H.: Efficient binary search for IP address lookup. IEEE Communications Letters, Vol. 9, No. 7, pp. 652--654. (2005).
[24] Lim, H., Lee, B., Kim, W. J.: Binary searches on multiple small trees for IP address lookup. IEEE Communications Letters, Vol. 9, No. 1, pp. 75--77. (2005).
[25] Lim, H., Kim,W., Lee, B.: Binary search in a balanced tree for IP address lookup. Proceding of IEEE HPSR2005, pp. 490--494. (2005).
[26] Wuu, L.C., Liu,T.J., Chen, K.M.: A longest prefix first search tree for IP lookup. Computer Networks, Vol. 51, Issue. 12, pp. 3354--3367. (2007).
[27] Lim, H., Kim, H. G.: IP address lookup for Internet routers using balanced binary search with prefix vector. IEEE Transactions on communications, Vol. 57, No. 3, pp. 618--621. (2009).

Authors

Hichem Haouassi holds the Engineer Diploma from the Computer Science department, HadjLAkhdhar University of Batna, Algeria in 2001, has received Magister and PHD degree on 2004 and 2012 from the Department of Computer Science, University of Batna, Algeria. He is a lecturer at currently a lecturer in the Department of Computer Science, University of Khenchela, Algeria. He is a member of Software engineering Group, at ICOSI Laboratory, ABBAS LAGROUR Khenchela University. His research interests include router architecture, mobile network, Data Mining and computational Intelligence.

ToufikMessaoud MAAROUK holds the engineer Diploma from the Computer Science department, University of ANNABA, Algeria; he obtained the degree of Magister in 2005 at Batna University in Computer science. And he obtained His PHD in 2012 at Constatnine University in Computer science he is an assistant Professor at Computer Science department of Khenchela University Algeria, where he teaches: Programming languages, graph theory, script languages and others matters. He supervises engineers and masters students on their final projects. He is a member of Formal methods Group, at ICOSI Laboratory, ABBAS LAGROUR Khenchela university,. He's research interests are in formal methods in programming and their applications, Artificial intelligence, emergent technologies.

Rafik MAHDAOUI holds the engineer Diploma from the Computer Science department, HadjLAkhdhar University of Batna, Algeria in 2001; he obtained the degree of Magister in 2008 at Batna University in Industrial engineering. And he obtained His PHD in 2013 at Batna University in Industrial engineering he is an assistant Professor at Computer Science department of Khenchela University Algeria, where he teaches: Programming languages, graph theory, script languages and others matters. He supervises engineers and masters students on their final projects. He is a member of computer security Group, at ICOSI Laboratory, ABBAS LAGROUR Khenchela university,. He's research interests are in Neuro-Fuzzy systems, Artificial intelligence, emergent technologies, prognosis and diagnosis,e-maintenance.