Proceedings of the Fifth IASTED International Conference
CIRCUITS, SIGNALS, AND SYSTEMS
July 2–4, 2007, Banff, Alberta, Canada
ISBN Hardcopy: 978-0-88986-669-0 / CD: 978-0-88986-670-6

CACHE REPLACEMENT POLICIES FOR IP ADDRESS LOOKUPS

Ruirui Guo,[+] José G. Delgado-Frias[+] and Stephan Wong[*]

[+]School of Electrical Eng. and Computer Sc.
Washington State University
Pullman, WA 99164-2752, USA
{rguo, jdelgado}@eecs.wsu.edu

[*]Computer Eng. Lab / Electrical Engineering
Delft University of Technology
2628 CD Delft, The Netherlands
J.S.S.M.Wong@ewi.tudelft.nl

ABSTRACT
In IP routing, the routing table lookup is a very important operation and the speed in which it is performed determines the overall performance of Internet processors. Consequently, caching schemes are implemented in such processors to speed up these operations. A well-known factor affecting the performance of caches is the replacement policy, which decides which entry in the cache to be replaced with a new one. In this paper, we propose two new cache replacement policies, namely LAR (Least Access and Recently used) and RLAI (Relatively Least Average Interval). With regard to the common LRU (Least Recently Used) replacement policy, our policies take an additional parameter into account to evict the relative inactivity of cache entries. We first evaluated both policies through extensive simulation based on IPv4 routing information; we have observed both of them improved the cache hit rate almost up to 5.73%, as compared with LRU. Then we extended to apply them on IPv6 routing, and it turns out that the LAR shows the hit rate increasing up to 2.35%, and the improvement of RLAI is up to 2.58% over LRU. Consequently the proposed replacement policies achieve better performance for both IPv4 and IPv6 routing.

KEY WORDS
Communication system, cache management, route lookup.

1. INTRODUCTION

IP routing plays an important role in the forwarding of packets through the Internet. It decides how and where to deliver incoming packets to the appropriate output interface (e.g., of a router) using the destination address. The process is performed by looking up entries in a routing table, which contains information relating to other networks and hosts on the Internet. Each entry in the routing table comprises an address prefix, a forwarding address, and the interface to which the packets should be delivered when their address prefix matches.

With the continuous growth of the Internet, higher demands are placed on IP routing in terms of speed; in particular, the growth of link bandwidths requires increasingly faster routing table lookups. A router needs to handle roughly 1,000 packets per second for every 10^6 bits per second of line rate [1]. Therefore, 10M routing table lookups per second are needed for a current route with the line rate of 10 Gb per second (OC-192). Moreover, in the near future, the line speed will grow towards 40 Gb/s (OC-768) with the continued technological advances in optical and electronic devices [2]. Such a high line speed requires fast table lookups to match. In order to speed up the address lookup process, a cache architecture [3] is generally used in network processors. A cache is a small memory with high memory access speed. The complete routing table is very large and stored in main memory with slow access time. Only those recently used forwarding information are held in cache. If a lot of forwarding information is found in cache, the average access time is reduced due to the small access count to the memory. There were also several schemes proposed to enhance caching effectiveness [4, 5, 6, 7]. By caching, if a destination address does not match any entry in cache, the whole routing table is searched, and then the cache is updated. In order to decide which entry in the cache should be replaced with a new one, a replacement policy needs to be used. In current cache organizations, two replacement policies are commonly used; namely Least Recently Used (LRU) [7] and Least Frequency Used (LFU). The LRU policy removes the entry that has the longest unaccessed time, while the LFU policy removes the entry with the least access count. In this paper, we propose two new policies, namely Least Access and Recently Used (LAR) and Relatively Least Average Interval (RLAI) policy, which help to achieve higher cache hit rates and in turn to make routing table lookups fast. The goal of these two policies is to evict the relatively inactive entry. An inactive entry is an entry that

has not been accessed for a relatively long time and, potentially, it will not be accessed in the near future. The potential future access is determined by a parameter introduced in the new replacement policies which is based on the history of the particular entry. The two policies were utilized in a simple fully associative cache while running both IPv4 and IPv6 [8, 9] traces respectively, and both of them result in higher cache hit rates. As we know, the current Internet protocol is IPv4 with the address of 32 bits, while IPv6 is the next generation protocol proposed as a long-term solution to solve the address limitations of IPv4 by providing a huge address space. In the future, it will gradually replace IPv4. Due to the long address format (128 bits) of IPv6, larger memory is required to store routing tables. This may restrict the lookup speed since a relatively large memory is used. This in turn creates a need for investigating a fast cache memory solution. The purpose of our novel replacement policies is to improve both IP routings without cost overhead.

This paper is organized as follows. In Section II, a cache architecture for address lookups is described, and the routing information are also briefly explained. Section III introduces our novel replacement policies. In Section IV, we present the simulation results using IPv4 traces and the extension results of IPv6 traces. Some concluding remarks are presented in Section V.

2. CACHE AND ROUTING INFORMATION

In this section we provide a brief description of the cache architecture that can be used for table lookup routing. We also describe the routing information used to simulate our replacement policies.

A. Cache architecture for IP routing

Cache memories are widely used in computer systems. In a program (with many memory accesses) ， it is very likely that the same data in memory is accessed multiple times in a short time period -temporal locality. Therefore, a small and fast memory can be used to reduce lengthy memory access times. Studies have shown that the network packet streams indeed have temporal routing locality. That is, a routing entry accessed before it is possibly referenced again in a short time. This characteristic allows caches to be used in IP address lookups. Consequently, more active forwarding entries are saved in cache; this in turn has the potential of making table lookups faster.

The process of lookups is associated with a destination address. When a router receives a packet from one of its interfaces, the destination address in this packet is extracted, and compared with the current entries in the cache using a Longest Prefix Matching mechanism [7]. The basic organization of the cache is depicted in Figure 1. If there is a matching entry in cache, this packet will be delivered to the interface specified by this entry. If a miss occurs, the address is searched in the large routing table stored in main memory. Subsequently, the matching entry is written into the cache. More importantly, a decision should be made on which entry in the cache to be replaced.

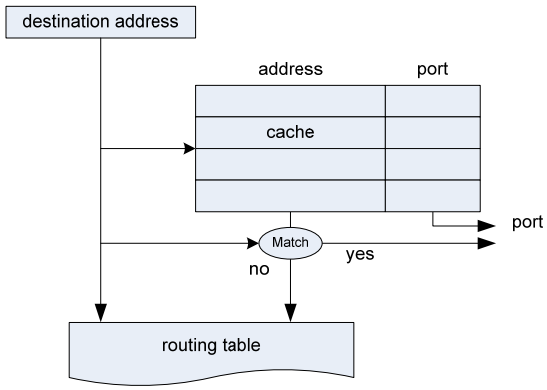

Fig. 1. A cache organization for IP address lookup.

Due to small size cache and temporal locality of destination addresses, this method helps to reduce the lookup time. The temporal locality helps to achieve a high hit rate. The average memory access time (AMAT) is calculated as follows [10].

$$AMAT=(H_{cache} X AT_{cache}) + ((1- H_{cache}) X AT_{memory}) \quad (1)$$

Where: H_{cache} is the cache hit rate, AT_{cache} is the cache access time, and AT_{memory} is the routing table access time.

As we mentioned before, the cache access time is smaller than the memory access time. Obviously, increasing the cache hit rate is beneficial to achieving smaller average access times.

B. Routing Information

Evaluations of our proposed replacement policies are based on three IPv4 and six IPv6 trace files which are downloaded from the Measurement and Analysis on the WIDE Internet (MAWI) Working Group http://tracer.csl.sony.co.jp/mawi/ [11]. They are all collected based on the daily traffic connected to some links. Each trace file has 2 million destination addresses. The length of each IPv6 address is 128 bits, which is four times longer than the current network address IPv4. Analyzing these incoming destination addresses in each trace file, we found that there are only several thousands unique addresses. This in turn indicates there exists temporal locality in these trace files and it supply the possibility of using a cache architecture. The information about these trace files is shown in Table 1. The trace file number is the one given at the website; we have labeled these files from X to Z for IPv4 and from A to F for IPv6 to ease file references.

We also download IPv4 routing tables from http://bgpview.6test.edu.cn/bgp-view/download.shtml [12], and obtain IPv6 routing tables both from the

machine "n6tap.es.net" on the website of 6TAP router information http://www.6tap.net/ [13] and from the machine "route-server.he.net" on Hurricane Electric Internet Services http://lg.he.net/cgi-bin/index.cgi [14] via telnet. Actually, since currently there are few users on IPv6, the sample IPv6 routing table size is not large. We have extended or combined tables to satisfy the unique destination addresses in the trace files.

Table 1. IPv4 and IPv6 Traces.

IP	Trace files	Number of unique addresses
IPv4	(X) 060524	70627
	(Y) 060609	76533
	(Z) 060615	63485
IPv6	(A) 031214	3949
	(B) 031215	3681
	(C) 031216	3422
	(D) 040128	3764
	(E) 040129	3874
	(F) 040130	3772

3. REPLACEMENT POLICIES

From Equation 1, a cache architecture requires a high cache hit rate to achieve a high performance. Since the main memory storing the routing table requires a larger access time than the cache, high hit rate reduces significantly the lookup time. An important factor effecting the hit rate is the replacement policy used when a conflict occurs. That is, when a fully associative cache is full and an entry from the routing table needs to be included, one of the current entries must be replaced with the new one. A replacement policy should help to maintain a high cache hit rate. In this section, two novel replacement policies are introduced; these policies yield a higher cache hit rate than the traditional LRU and LFU policies which are presented below.

A. Existing Replacement Policies.

The following policies are two of the most commonly used replacement policies in cache systems. They are briefly described below.

Least Recently Used (LRU) policy: this policy keeps a list of entries that are currently in the cache. When an entry is accessed, this entry is moved to the front of the list. When a miss occurs and the cache is full, a replacement is needed; the entry at the bottom of the list is removed. The new entry found in memory is placed into the cache and the list is updated. Simply said, the LRU policy evicts the entry that has not been accessed for the longest time.

Least Frequently Used (LFU) policy: this policy evicts the entry that has been the least frequently used. The motivation behind this policy is that some entries are accessed more frequently than others in a given time. This policy sets an access counter as an estimate of the frequency. The entry with the lowest access count is removed from the cache.

B. Proposed Replacement Policies.

We have developed two novel replacement policies, which involve more than one parameter to make an eviction decision. In order to make the following description concise, we use IPv6 routing to illustrate these two policies.

Least Access and Recently used (LAR) policy. The goal of this policy is to evict the relatively inactive entry. By analyzing IPv6 traces, we found an interesting case that some entries just evicted from cache by either LRU or LFU policies tends to be re-accessed in a short time later. Table 2 shows the statistic information about such entries in trace (A). That is, collecting the number of the route entries which are evicted based on LRU policy because of cache space, but actually required to be re-accessed within next M lookups. This table shows different number of cache entries.

Table 2. The Number of Re-accessed Evicted Entries for Trace (A)031214.

Num. of cache entries	Num. of evicted entries	Number of lookups after eviction (M)				
		0-50	51-100	101-200	201-300	301-400
64	587968	69174	52114	64108	40614	30475
128	369922	18147	14972	24714	19681	17223
256	205667	4782	4297	7673	6612	6142

As for a given 64-entry cache, more than 69,000 entries among all the 587,968 entries are evicted but are re-accessed within the next 50 lookups. Over 50,000 evicted entries are re-accessed between 51 and 100 lookups. Evicting entries that are accessed within a small number of lookups decreases the cache efficiency. LRU or LFU have no means to address this issue since they are implemented solely based on one aspect of the past activity of entries in the cache. One is based on the unaccessed time, and the other is on the access count. If we take into account both aspects, and choose the inactive entry in cache, this will increase the hit rate. This in turn decreases the AMAT.

In Figure 2, list A is a group of cache entries sorted by unaccessed time; this is the same as that of the LRU policy. The recently accessed entry B is put in the top of list. The LAR replacement policy removes the entry with the lowest access count among the bottom N recently unaccessed entries in the list. In the example, entry C is the candidate to be evicted by the LAR policy due to its lowest access count. However, if N is close to the number of the cache entries, and the candidate happens to be accessed recently, and then we evict the least recently used entry instead of this candidate. That is, the evicted entry by LAR is the result of considering both access time and access count.

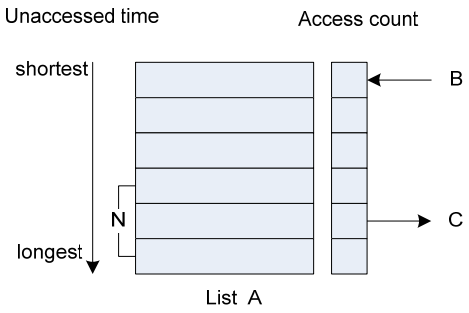

Fig. 2. LAR Replacement Policy.

The value of N could be optimized by analyzing the cache performance under different N. Table 3 shows the hit rate increments of LAR over LRU policy for different N with a 128-entry cache organization. Figure 3 shows that for N equal to 32 (1/4), the LAR policy achieves the best performance on the sample IPv6 traces. For the traces we have studied the value of N that yields the highest hit rate is 1/4.

Table 3. Cache Hit Rate Increment of LAR over LRU with Different Size N (using a 128-entry cache).

Trace	N							
	1/8	1/4	3/8	1/2	5/8	3/4	7/8	1
A	1.22	1.82	1.79	1.29	.98	.54	0	0
B	1.05	1.61	1.53	.98	.76	.46	0	0
C	.92	1.27	1.08	.87	.81	.58	0	0
D	.53	.68	0	0	0	0	0	0
E	.51	.83	.55	0	0	0	0	0
F	.55	.96	.99	.83	.37	.11	.02	.02

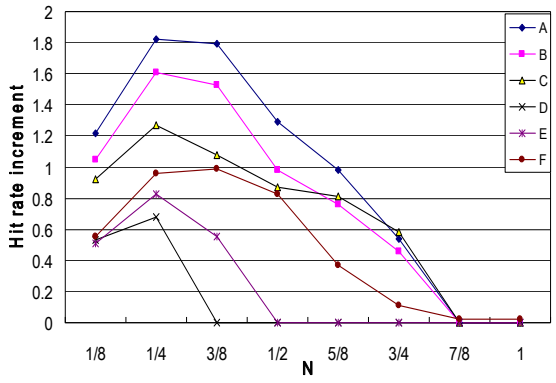

Fig. 3. Plot of Hit Rate Increments of LAR over LRU with Different N,

Relatively Least Average Interval (RLAI) policy. The prime motivation for this policy is similar to LAR, but a different parameter is considered when making replacement decisions. An array is used to save the average interval between two accesses of the same entry. This interval is determined by counting the number of lookups between such two accesses. Each time the entry is accessed a new average interval is computed. If an entry is not accessed for a time larger than its average interval, then it is considered as an inactive entry and it has the potential to be evicted. This algorithm chooses an entry with the longest average unaccessed time among these entries potential evicted. The implementation is depicted in Figure 4.

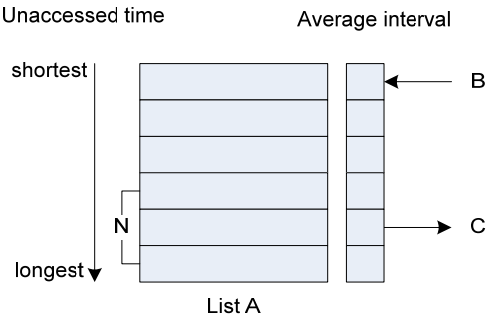

Fig. 4. RLAI Replacement Policy

These replacement policies can be implemented using a simple linked list in hardware. This list needs few entries with the size of the optimum N of the total cache. These entries are arranged in the order of access count or average access interval.

4. PERFORMANCE EVALUATION

In this section, we implement our proposed replacement policies on cache architecture, and compare their performance to the LRU policy; we first use IPv4 destination address traces to evaluate the cache performance, and then extended to apply these replacement policies on IPv6 traces to indicate that the policies proposed also contribute to this new protocol.

In our evaluation, we generally choose hit rate as the measurement of performance for cache architecture with different replacement policies. Hit rate provides a good indication of the potential performance for cache schemes. Because memory access time is much longer than cache access time, many searches in memory will slow down the lookup process. However, with a high hit rate, fewer accesses to the routing table in the main memory are needed. We do not include the LFU policy here, because its performance is worse than that of LRU in all the traces. For example, the hit rate of LFU on the trace (B) 031215 is less than that of the LRU by up to 19 % depending on different cache sizes.

A. IPv4 Performance Evaluation

The following Table 4 shows the cache hit rate of the three replacement policies, including the two novel policies we introduces with different number of cache entries. Figure 5 indicates that the hit rate changes with the increasing cache entries for a given sample trace (X) 060524.

Table 4. Hit Rate with Different Replacement Policies.

Num.of cache entries	Policy	IPv4 traces		
		X	Y	Z
512	LRU	73.94	69.12	74.76
	LAR	78.11	74.60	79.51
	RLAI	78.26	74.85	79.70
1K	LRU	83.98	81.45	85.43
	LAR	85.58	82.52	87.10
	RLAI	85.54	82.32	87.10
2K	LRU	89.53	88.04	91.29
	LAR	90.10	88.85	91.72
	RLAI	90.04	88.81	91.70
4K	LRU	92.71	91.88	94.12
	LAR	92.89	92.41	94.35
	RLAI	92.84	92.40	94.32

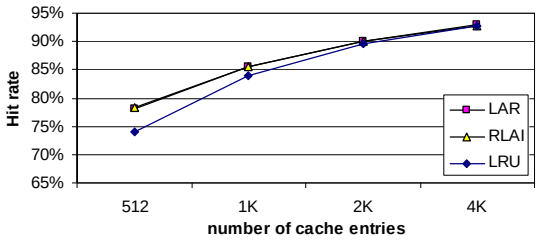

Fig. 5. Cache Hit Rate Performance of Trace (X).

From the above information, it can be observed that both LAR and LRU policy have a better performance than the common LRU. If only LRU is applied, only one parameter is considered in replacing an entry. This entry, however, may be needed shortly after it has been removed. When using our proposed policies, this problem is alleviated. Our simulation results show that the hit rate is increased from 0.18% to 5.48% for LAR and from 0.13% to 5.73% for RLAI, depending on the trace files and the number of cache entries.

Table 5 shows another aspect of cache performance, i.e. increment ratio, which is the ratio of the hit rate increment by novel replacement policy to the hit rate increment by doubling the number of cache entries. Obviously, without any change in cache size, our proposed policies can achieve up to 46% increment performance by comparing with doubling the cache. Although the hit rate improves as the cache size doubles, the expense on caches also increases, especially for expensive fully associative cache. Thus our replacement policies are of advantage to enhance cache performance without the overhead cost.

Table 5. Increment Ratio of Cache Performance.

Num. of cache entries	Policy	IPv4 traces		
		X	Y	Z
512	LAR	41.51	44.50	44.51
	RLAI	43.01	46.49	46.31
1K	LAR	28.72	16.37	28.50
	RLAI	28.07	13.26	28.53
2K	LAR	17.82	21.19	15.10
	RLAI	16.01	20.02	14.36

Obviously, the LAR and RLAI policy are actually good at achieving higher hit rate with regard to IPv4 routing information. We will show below both of them keep effective when extending to IPv6 routing lookups, which is important for Internet development in future.

B. IPv6 Performance Evaluation

By applying these two replacement policies LAR and RLAI, the cache hit rate with different number of cache entries when processing six IPv6 traces is explained in Table 6, Figure 6 depicts the different hit rates for a 128-entry cache.

Table 6. Hit Rate with Different Replacement Policies.

Num.of cache entries	Policy	IPv6 Traces					
		A	B	C	D	E	F
64	LRU	70.60	75.70	76.63	80.34	81.64	84.01
	LAR	72.27	76.95	77.89	81.36	82.34	84.18
	RLAI	73.01	77.64	78.59	81.90	82.93	84.84
128	LRU	81.50	84.53	85.26	88.49	88.96	89.85
	LAR	83.31	86.14	86.53	89.17	89.79	90.84
	RLAI	83.85	86.57	87.12	89.90	90.30	91.27
256	LRU	89.70	91.18	92.13	93.88	94.14	94.83
	LAR	92.05	93.07	93.56	95.04	95.16	95.61
	RLAI	92.28	93.48	94.02	95.27	95.29	95.77

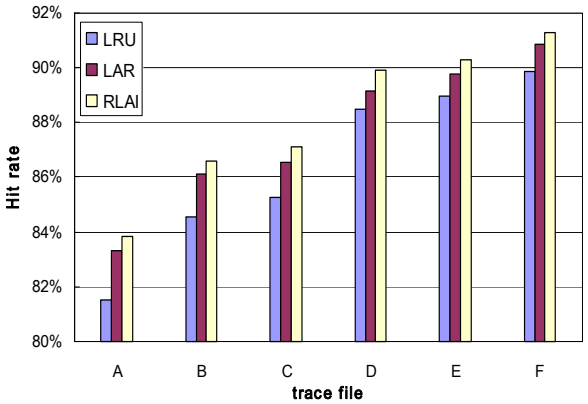

Fig. 6. The Performance of Cache with 128 entries.

Similar to IPv4 routing, our simulation results of IPv6 show that the hit rate is increased from 0.17% to 2.35% for LAR and from 0.83% to 2.58% for RLAI, depending on the trace files and the number of cache entries. Actually, LAR and RLAI yield over thirty thousand additional matches (or hits) as compared to LRU.

Figure 7 depicts the hit rate with the increasing cache size (number of entries) on the trace (A) 031214. As we mentioned before, since the cache cost must be increasing with the increasing cache size, our replacement policies help to enhance cache performance. The RLAI is a little better than the LAR, and both of them have higher hit rate than the LRU policy for different cache sizes. The improvement rate diminishes as the cache size grows. The higher hit rate in turn is translated into higher speedup of the routing table lookup.

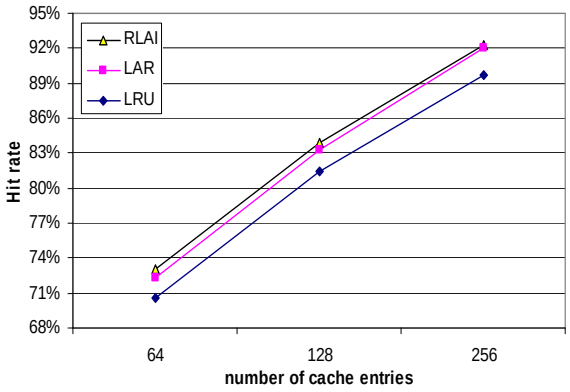

Fig.7. Hit Rate with Increasing Cache Entries.

5. CONCLUDING REMARKS

In a cache scheme, a replacement policy is used to make a decision about which entry in cache should be removed to allow a new one to enter. This policy affects the cache performance. In order to increase the cache hit rate for IP address lookup, we have proposed two novel replacement policies and evaluated their impact. We compared these policies with the commonly used *Least Recently Used (LRU) policy*. The simulations show our proposed policies achieve better performance on both IPv4 and IPv6 routing. The features of our replacement policies are as follows.

- *Least Access and Recently used (LAR) improves hit rate from 0.18% to 5.48% for IPv4 and from 0.17% to 2.35% for IPv6 over LRU, respectively*. It makes a removing decision based on two major parameters: unaccessed time and access count. It evicts the entry with the smallest access count among N entries that have not been accessed for a long while.
- *Relatively Least Average Interval (RLAI) policy improves hit rate from 0.13% to 5.73% for IPv4 and from 0.83% to 2.58% for IPv6 over LRU, respectively.* It uses the average interval between two accesses of one entry as a factor to make a decision. If an entry has not been referred longer than its previous access interval, it can be potentially evicted.
- *Our newly introduced policies tend to evict more inactive entries*. Both policies tend to remove an inactive entry by evaluating its previous access references, which include unaccessed time and access count or unaccessed time and average access interval.
- *Alleviating required access of recently evicted entries.* LRU performs evictions depending only on unaccessed time. An entry evicted from cache may be referred in a short period of time after its eviction; this is a larger problem for small caches. Our policies can alleviate such situation because they evaluate one entry based on multi-conditions, and they tend to keep more active entries in the cache to reduce the number of the replacements.

A replacement policy is a way to improve caching effectiveness. A good policy is able to increase the hit rate. A high hit rate makes average memory access time fast and in turn speeds up the address lookup process.

REFERENCES

[1] P. Gupta, S. Lin, and N. McKeown, Routing lookups in hardware at memory access speeds. *Proceedings of InfoCom'98*, San Francisco, CA, April 1998, 1240-1247.

[2] H. Chao, Next generation routers. *Proceedings of The IEEE, Vol. 90, No. 9*, September 2002, 1518-1558.

[3] D. Feldmeier, Improving gateway performance with a routing-table cache. *Proceedings of InfoCom'88*, March 1988, 298-307.

[4] B. Talbot, T. Sherwood, and B. Lin, IP caching for terabit speed routers. *Proceedings of Globecom.1999*, 1565-1569.

[5] T. C. Chiueh and P. Pradhan, Cache memory design for network processors. *Proceedings of High-Performance Computer Architecture*, January 2000, 409-418.

[6] R. Guo and J. G. Delgado-Frias, A pipelined cache architecture for IPv6 lookups. *International Conference on Communication and Computer Networks (CCN 2004)*, Cambridge, Mass., November, 2004.

[7] J. J. Ronney, J. G. Delgado-Frias, and D. H. Summerville, An associative ternary cache for IP routing. *Int. Conf. on Communications and Computer Networks*, Cambridge, MA, November 2002.

[8] C. Huitema, *IPv6: the New Internet Protocol*, 2nd edition, Prentice Hall, 1998.

[9] RFC 2373, IP Version 6 Addressing Architecture. July 1998.

[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd edition, (San Francisco: Morgan Kaufmann Publishers Inc. 2003).

[11] The Measurement and Analysis on the WIDE Internet (MAWI) Working Group. http://tracer.csl.sony.co.jp/mawi/

[12] CERNET BGP VIEW Global Internet. http://bgpview.6test.edu.cn/bgp-view/download.shtml

[13] 6TAP router information. http://www.6tap.net/

[14] Hurricane Electric Internet Services. http://lg.he.net/cgi-bin/index.cgi