AN ASSOCIATIVE TERNARY CACHE FOR IP ROUTING

James J. Rooney[1] José G. Delgado-Frias[2] Douglas H. Summerville[1]

[1]Department of Electrical and Computer Engineering
State University of New York
Binghamton, NY 13903-6000

[2]School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752

Abstract. In this paper we present a study of a prefix routing cache for Internet IP routing. An output port assignment requires one cache memory access when the assignment is found in cache. The cache array is divided into sets that are of variable size; all entries within a set have the same prefix size. The cache is based on a ternary content addressable memory (TCAM) that matches 1, 0 and don't care (X) values. Our study shows that an associative ternary cache provides an output port at the speed of one memory access with a very high hit rate. For an 8K-entry cache the hit rate ranges from 97.62% to 99.67% on traces of 0.2 to 3.5 million addresses. A port error occurs when the port selected by the cache differs from the port that would have been selected from the routing table. In this paper we introduce a novel sampling technique that reduces the worst port error rate by an order of magnitude (from 0.52% to 0.05%).

1. Introduction

The continuous growth in Internet related technologies and applications have imposed higher demands on network routers [12]. This in turn has led to the emergence of network processors that are designed to execute network protocols at different computer network layers. These layers include: the link switches –layer 2, network routers –layer 3, and transport gateways –layer 4 [1]. Some of the specific tasks include: *Routing* that determines the output port to forward packets; *Forwarding* that sends packet according to a flow control strategy; *Scheduling* which schedules packets according to their priorities to support Quality of Service –QoS; *Filtering* that monitors packets to support privacy –firewall; and *message manipulation* that supports application specific processing and time-to-live counter.

With the increasing speed of communication channels between computer and other devices in a network, it is important that the routing algorithm execution time be extremely short. This time has a great influence on how fast a packet advances through the network. A packet cannot be transferred until the next hop is selected. Routing algorithm execution time must be reduced to decrease packet latency. The basic transfer unit on the Internet is the Internet datagram, which is divided into header and data areas. The datagram header contains the source and destination addresses. Each host on the Internet is assigned a unique 32-bit Internet address that is used in all communication with that host. The IP address consists of two parts: the network part and the host part. The network part identifies a network while the host part identifies a host on that network. IP Routing does not alter the source and destination addresses of the original datagram.

Each IP routing table entry includes the following fields: a network mask, a destination network address and a next-hop address. In IPv4, the network mask and destination address fields are 32 bits long. The network mask field is composed of contiguous binary 1's to designate the network field followed by binary 0's to fill out the 32-bit field. Given an incoming packet's destination address, the network mask field of an IP routing table entry is used to extract the destination network address, which is then compared to the entry's destination network address field. If they match, this routing table entry is considered a potential candidate. This process is repeated until all possible addresses in the routing table have been considered. When there are multiple matches the match with the longest network address is chosen over the others; this is called longest prefix match [3]. The packet is routed via the output port

specified in this entry to the corresponding next-hop router. If none of the table entries match the incoming IP packet's destination host address, the packet is forwarded to the default router. To reduce time in executing best matching prefix algorithms the search is typically done using a tree-like structure. Accessing this structure requires a number of memory references; this number depends on how deep the tree is. A number of hardware schemes have been proposed to speed up IP lookups. These schemes include TCAM (ternary content addressable memory) [8], a pipelined DRAM memory scheme that achieves table lookup in one memory cycle [4], and an IP address cache, which achieves lookup in one CPU cycle [1]. Only the TCAM schemes aim to provide O(1) lookup speeds. The proposed approach is based on a TCAM scheme that features a parallel search of all entries, thereby performing a cache lookup in one cache memory cycle. The TCAM approach proposed by Liu [8] also achieves lookup in one cache memory cycle at the expense of increasing the routing table size significantly. Some of these alternative approaches are discussed in more detail in Section 4.

This paper has been organized as follows. In Section 2, the associative cache routing scheme is described. The scheme's performance is presented in Section 3. A comparison of our results to related work is presented in section 4. Some concluding remarks are provided in Section 5.

2. Associative Cache Scheme

The proposed approach is based on an IP prefix cache using a high performance bit-pattern associative array system. The associative router implements a Classless Inter-Domain Routing (CIDR), which is the most common and flexible form of Internet routing [13]. Since the data streams presented to Internet Routers exhibit different characteristics than general purpose CPUs, our cache scheme is considerably different than a conventional CPU cache. We have applied the basic Bit-Pattern Associative Array [14] approach to be used as the IP prefix cache for Internet routing tables. The TCAM based approach features a parallel search of all entries, thereby performing a cache lookup in one cache memory cycle.

2.1 Bit-Pattern Cache for IP Routing

Figure 1 shows the proposed bit-pattern scheme for cache Internet routing lookups. The cache array is divided into 32 sets, one set for each of the prefix sizes that are possible with a 32-bit IP address. The set numbers used in this paper are the same as the suffix sizes, not the prefix sizes; set 0 contains the entries with a prefix size of 32 and a suffix size of 0. The cache resources are divided into virtual sets whose size can be allocated statically to those sets that need them. Routing tables have a history of being heavily skewed to entries with suffix sizes of 8 to 16 (prefix size 16 to 24). For these simulations the cache set sizes have been made proportional to the corresponding prefix size in the routing table. All entries within a set have the same suffix/prefix size. This scheme allows the cache to have a natural priority order; sets are arranged with the longest prefix (set [0]) on top and the shortest prefix (set [31]) on the bottom.

Each cache entry contains a 32-bit network address and a designated port. Thus, each entry corresponds to and is equivalent to one routing table entry. The cache network address entry is generated by masking the standard route table network address with the network mask, and replacing all low order elements below the prefix-suffix boundary with don't care elements. For example, a typical CIDR routing table entry of IP address, mask and port is: 9.20/255.255.128 IBM (port designation)

In the bit-pattern associative scheme, this entry would be converted to a single 32-bit address and the designated port as follows. 00001001 00010100 0XXXXXXX XXXXXXXX IBM(port)

The operation of the proposed Bit-pattern Associative Cache (BPAC) approach is described below.

1) *Destination Address.* A 32-bit destination serves as input to the cache array.
2) *Associative Pattern Array.* As shown in Figure 1 the addresses in the bit-pattern array are grouped into sets, based on the prefix length of the network address. The upper set in the cache is set [0] with a

32-bit prefix and proceeding down to bottom of the array, we find set [31] with a one bit-prefix. An implication of this is that the array is ordered, starting at the top by the longest network addresses. When searching for a match the destination address is compared to all entries of the array in parallel. The match closest to the top of the array (set [0]) has the longest network address.

3) *Priority.* For a basic IPv4 router, since the top-most match would have the longest network address, the port from that entry would be selected. Additional services can be performed by adding logic in this function. For instance filtering, which is the denial of certain networks from designated sources, and priority based routing could be accomplished in this selection function. If alternative ports are available to select from this could also be handled in this priority function. A high performance priority encoder design can be found in [2].

4) *Output interface assignment.* Selecting the output port associated with the selection condition makes the output port assignment.

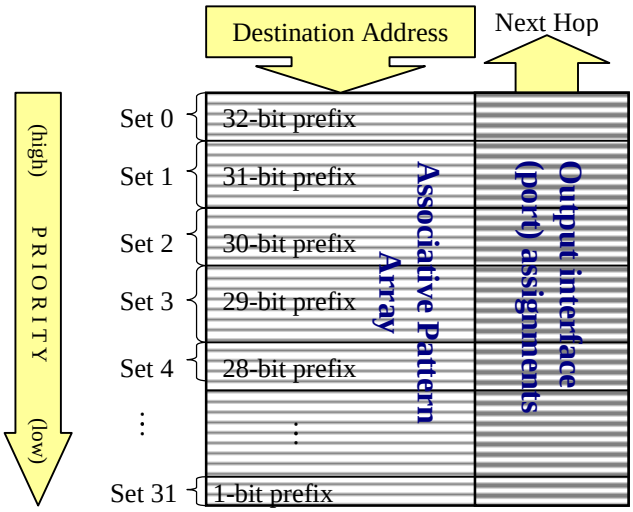

Figure 1. Bit-pattern associative scheme.

2.2 Ternary Content Addressable Memory

Ternary CAMs are beginning to have acceptance as a viable alternatives for network processors [5,6,7]. This type of CAM is able to match in parallel stored data with incoming data, which in this case is a destination address. The stored data have values composed of 0, 1, and don't care (X). Figure 2 shows an example of a CAM with 8 32-bit entries with two sets. If the following destination address is presented `10101110 11100001 01010001 01000110`; this would match the last entry of set 8 as well as the second entry of set 12. The entry on set 8 should be chosen to implement longest prefix routing.

Set 8	10101110 11100011 00111000 XXXXXXXX
	10101110 11100001 01010111 XXXXXXXX
	10101110 11100001 01010001 XXXXXXXX
Set 12	10101110 11000011 0001XXXX XXXXXXXX
	10101110 11100001 0101XXXX XXXXXXXX
	11001100 00001100 1001XXXX XXXXXXXX
	10001100 00001100 1010XXXX XXXXXXXX
	00001100 00001100 0001XXXX XXXXXXXX

Figure 2. Example of ternary CAM entries.

Using a ternary CAM as the pattern associative array, the don't care bits are on the right side of each entry. The set number indicates the number of don't care bits per entry.

2.3 Bit-pattern Associative Cache Organization

An organization for the proposed associative cache is shown in Figure 3. The associative cache uses a ternary content addressable memory as its associative pattern array, and this enables destination address alternatives to be considered in parallel. The destination address is presented as the input to the ternary CAM array for comparison with the stored data. In the figure two patterns or entries match the input. These results are passed to a selection function unit (which is a priority encoder). The encoder passes the match output with the highest priority to the port assignment to select the word corresponding to the selected output port.

The major components of the cache organization are briefly described below.

- *TCAM Array:* This unit stores the associative patterns and performs a comparison between the destination address and routing table. All the patterns that match the input address are passed to the priority encoder.
- *Priority Encoder:* This unit selects only one of the outputs from the CAM. If more than one pattern has been matched, the one with the highest priority gets selected. The match closest to the top of the array has the longest prefix size and therefore the highest priority.
- *RAM:* This is a memory that stores the output port assignments. The address where the assignment is read is specified by a pointer from the priority encoder
- *Decoders:* These decoders are used as pointers to a row of the CAM or RAM arrays to write onto specific locations.

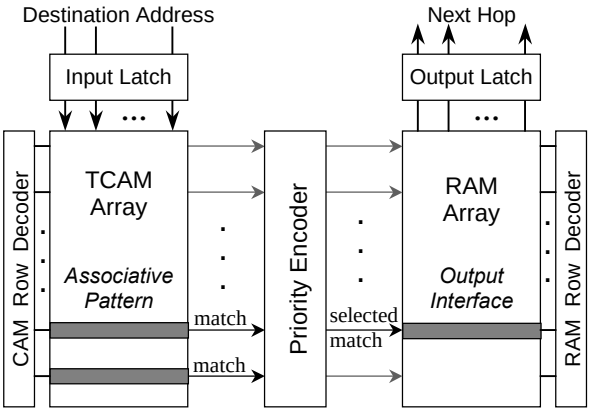

Figure 3. Associative ternary cache organization.

The associative ternary cache organization can be pipelined to improve throughput and reduce clock cycle time. Three stages can be implemented (in parenthesis is the associated hardware): Associative search (ternary CAM), selection stage (priority encoder), and assignment stage (RAM).

3. Associative Ternary Cache Performance

We have studied the performance of our associative router using six IP header trace files, which have been obtained from the National Laboratory for Applied Network Research (NLANR) web site http://pma.nlanr.net/PMA/ [10]. Files in these directories contain public packet header traces of Internet traffic with sanitized IP address fields. The encoding used to sanitize the IP address fields, however, is done in a manner to retain all statistical features of the original fields. To make these files useful, a set of

IP addresses were obtained from the routing table, and used as replacements for the sanitized address fields. Sanitized addresses that were repeated use repeat addresses from the routing table. The resulting trace files maintain the statistical features of the original traces. This assured that the temporal locality of the IP Header files was maintained. Six IP packet traces, containing 12,450,768 header files were downloaded and used to predict the cache performance. Our routing tables were obtained from the web site: http://www.merit.edu/ipma [9]. In all cases our simulations consider only one port selection for each routing table entry. If a routing table entry had options for the port selection, we use the first option. The routing table used in our simulation was a snapshot dated 01.10.01 from a mae-west router and had 32019 unique entries.

A summary of the IP Address Trace Files (from http://pma.nlanr.net/PMA/) and their features are shown in Table 1. The first column identifies the file in the pma.nlanr.net web site directory. The second and third columns indicate the number of IP Headers and unique IP destination addresses, respectively. The ratio of number of packets to number of unique addresses could be used as an indicator of temporal locality in a trace file. This ratio is shown in the last column. Notice that the maximum number of unique address in any one-trace file is just below the number of entries in the routing table. If they exceeded the number of entries (32019) we would have generated more than one IP address entry from the required routing table entries.

Table 1. IP address trace files obtained from [10].

IP Header File ID	Number of packets dest. addresses	Number of unique IP addresses	ratio
20011206/ADV-1007598319-1	203,352	1,465	139
20011001/BWY-1001908207-1	1,774,231	5,690	312
20011207/TAU-1007728734	1,816,318	9,601	189
20011001/BWY-1001895342-1	1,887,643	17,119	110
20011001/IND-1001895343-1	3,266,998	32,014	102
20011206/APN-1007673284	3,502,216	15,264	229

3.1 Cache hit rate

One of the prime measures of performance on caches is hit rate. This refers to the percent of cache accesses that successfully retrieve the data. In our case the data would be the output port. The principle of locality helps to achieve a high cache rate in microprocessors. This principle (temporal locality) is present in most routers; i.e. packets with the same destinations tend to cluster in time. Our simulations used 1K, 2K, 4K, and 8K caches of variable size sets. For these simulations, the number of entries assigned to the caches is made proportional to the entries in the routing table. An example of a 4K cache is shown in Table 2. Sets 0 to 7 and 20 to 24 have one entry per set; set 8 has the largest number of entries.

Table 2. 4K variable set cache entries by set (4096 entries).

Set	Number of entries	Set	Number of entries
0–7	1	15	71
8	2075	16	332
9	343	17	20
10	299	18	10
11	215	19	8
12	260	20–24	1
13	327	25–31	0
14	123	-	-

We have run a number of simulations with traces that contain from 0.2 to 3.5 million routing decisions. Table 3 shows cache-hit rates (CHR) for 1K, 2K, 4K, and 8K-entry caches. Cache hit rates range from 70.84% (3.26M trace, 1K cache) to 99.67% (1.77M trace, 8K cache). The trace of 3.26 million packets exhibits less locality than any other; this in turn leads to lower hit rate. It should be noticed that the hit rate for this trace greatly improves as the cache size increases. For all the other traces, the hit rate is very high and improves with cache size.

Table 3. Cache hit rate (1K, 2K, 4K and 8K entries)

Trace size	1K	2K	4K	8K
203,352	98.91%	99.20%	99.20%	99.20%
1,774,231	92.50%	99.44%	99.65%	99.67%
1,816,318	94.96%	98.76%	99.35%	99.49%
1,887,643	92.75%	98.34%	98.99%	99.10%
3,266,998	70.84%	82.15%	93.21%	97.62%
3,502,216	81.83%	97.22%	99.26%	99.54%

Figure 4 shows cache hit rate and its relation to the ratio of the number of packets and unique IP addresses. It should be noted that benchmarks with lower ratios tend to have lower cache rate, for cache sizes greater than 1K. This is due to limited temporal locality.

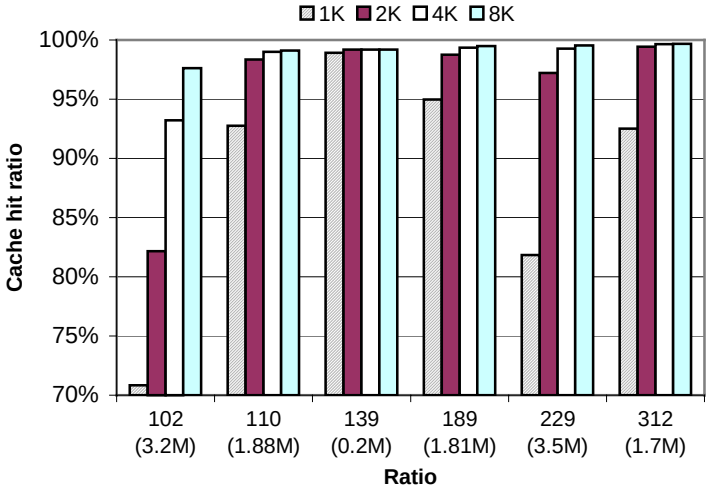

Figure 4. Plot of cache hit rate and number of packets/unique IP addresses ratio

3.2 Port error rate

A port error occurs when the port selected by the cache differs from the port that would have been selected from the routing table. This in turn may lead to an inappropriate port assignment. Table 4 presents the simulation results of the traces' port error rates.

Table 4. Port error rate (1K, 2K, 4K and 8K-entry)

Trace size	1K	2K	4K	8K
203,352	0.023%	0.012%	0.012%	0.012%
1,774,231	0.062%	0.029%	0.026%	0.026%
1,816,318	0.316%	0.031%	0.055%	0.048%
1,887,643	0.193%	0.072%	0.071%	0.084%
3,266,998	0.202%	0.327%	0.482%	0.520%
3,502,216	0.114%	0.090%	0.086%	0.097%

From Table 4, it can be observed that the port error rate is extremely small in particular for the 8K-entry cache. There is not a specific pattern to this measurement other than the error rate is decreased for cache sizes above 1K except for the 3.2M traces, which has limited locality. Port errors result in the routing of packets on a non-optimal path, but unless a packets time out is exceeded the packet will be delivered to the correct destination. If a packet's time out is approached it should be passed directly to the routing table to avoid any more non-optimal routing. Port errors are analogous to misrouting caused by routing table changes that occur in all IP networks when a particular node is not aware of a change that is propagating in the network this node will send the packet on a non-optimal path.

The major causes for port misses are:

Match to a wrong entry. The longest prefix network address match, for the destination address being searched is in the routing table but does not happen to be in the cache at this time. The cache finds another match, and in some cases the port assignment is not the same as would be found in the routing table.

Cache coherency. In this case, the problem is that changes to the routing table are made frequently, making the port assignment different in the routing table and cache.

Cache coherency can be assured by updating the cache with the new information that is received by the routing table or by flushing the cache every time the routing table is updated. Another approach that would reduce port misses from either of the two causes would be to use a sampling technique, where samples of cache assignments are compared to the routing table assignments for confirmation.

3.3 Port error rate improvements with sampling

Our sampling technique requires a search of the routing table after a cache hit and a comparison of the ports selected by both searches. When a port selected by the cache is different than that selected by the router, the cache entry is removed and the routing table entry is written to the cache. The port selection on all cache hits, even if they are in error, are returned to the cache controller. The search of the routing table after a cache hit is performed at a sampling rate designed into the cache simulator software. Table 5 presents the simulation results of the port error rates at a sampling rate of 33%. At this rate the routing table is searched every third cache hit and the port selections compared. The average reduction of port misses in these simulations is close to an order of magnitude when measured against all 6 traces and the 4 cache sizes. It should be noted that the patterns remain the same as Table 4, namely that port misses decrease with cache size greater than 1K with the exception of the one trace that shows poor locality.

Table 5. Port error rate with sampling.

Trace size	1K	2K	4K	8K
203,352	0.003%	0.003%	0.003%	0.003%
1,774,231	0.017%	0.002%	0.002%	0.002%
1,816,318	0.011%	0.005%	0.005%	0.006%
1,887,643	0.019%	0.007%	0.009%	0.011%
3,266,998	0.052%	0.081%	0.066%	0.050%
3,502,216	0.044%	0.006%	0.006%	0.008%

It should be pointed out that all the IP traces have an improvement that ranges from 2.59% (3.5 IP trace, 1K cache size) to 28.7% (1.81 IP trace, 1K cache size). This improvement is obtained by having port error rate with no sampling divided by port error rate with sampling.

The cache-hit ratio did not change significantly when port error sampling is introduced. The largest changes occurred with the 3.27M trace file where cache-hit ratio changed from 93.21% to 93.05% for the 4K cache. Variations in the sampling rate were simulated on an 8K cache with the 1.77 trace. The

improvement with 33% sampling of 16 is reduced to an improvement of 11 with 20% sampling and 5 with 10% sampling.

3.4 Cache writes

Another issue to consider in our scheme would be the number of times the cache needs to be written. Table 6 shows the number of writes that are required to handle the traces and the cache sizes. In Figure 5, we show the number of writes relative to the 1K cache writes. It should be pointed out that the number of cache writes decreases as cache size increases as shown in Tables 5. For large traces the reduction in the number of writes approaches an order of magnitude.

Table 6. Number of cache writes.

Trace Size	1K	2K	4K	8K
203,352	1792	1624	1624	1624
1,774,231	87001	9994	6205	5809
1,816,318	73647	22600	11866	9378
1,887,643	130346	26979	19160	17248
3,266,998	937696	585927	227005	81040
3,502,216	613931	97861	26108	16198

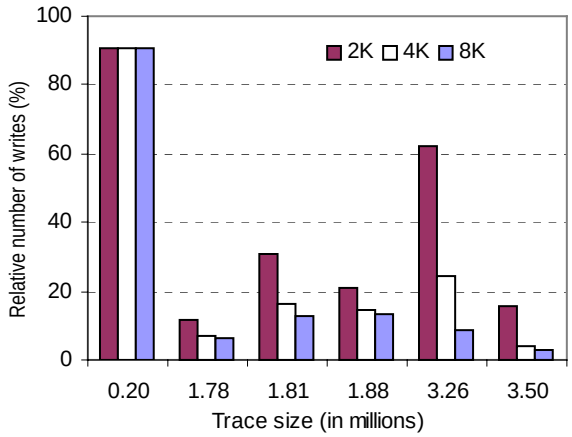

Figure 5. Relative number of cache writes.

Additional cache writes are required when our sampling technique is used to reduce port errors. The additional cache writes due to sampling (shown in Table 7) is small relative to the total cache writes amounting to a 0.65% additional writes in the worst case (8K cache with 3,226,998-trace file). The additional cache writes for each benchmark and cache size is shown as a percent increase in total cache writes in Figure 6.

Table 7. Cache Writes due to 33% sampling.

Trace Size	1K	2K	4K	8K
203,352	3	2	2	2
1,774,231	88	15	11	9
1,816,318	57	28	34	36
1,887,643	110	43	49	65
3,226,998	386	703	662	524
3,502,216	406	72	90	97

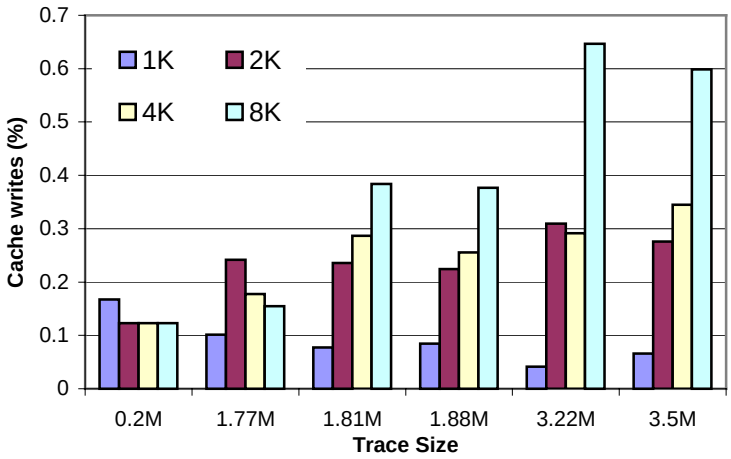

Figure 6. Percent of cache writes due to sampling.

3.5 Replacement policies

When a set is full and a new entry needs to be inserted a replacement has to be made. We have investigated the impact of the replacement policies on the performance of our approach. Three primary cache replacement strategies are employed to select the cache entry to replace. These are: Random—candidate entries are randomly selected; Least-Recently Used (LRU)—the entry that is replaced is the one that has been unused for the longest time; and First In First Out (FIFO)—the first cache entry written to a set will be the first replaced. Table 8 shows the result of 1.78M benchmark simulation on a 1K cache without port error rate sampling.

In all of the benchmarks, the cache-hit ratio with LRU performs better than the other two replacement policies, generally achieving an improvement of approximately 2% in cache hit ratio and a 37% reduction in cache miss ratio. The cache miss ratio is defined as 100% minus the cache hit ratio.

Table 8. Performance of replacement policies.

Measurements	LRU	Random	FIFO
Cache Hits	1689616	1640595	1640325
Port Hits	1682882	1635470	1635693
Port Misses	6734	5125	4632
Cache Writes	84615	133636	133906
Cache Hit Ratio	95.23%	92.47%	92.45%
Cache Miss Ratio	4.77%	7.53%	7.55%

4. Comparison with Other Schemes

Related studies have been reported by C. Partridge of BBN [11], T. Chiueh and P. Pradhan of SUNY-Stony Brook [1] and H. Liu of Stanford University [8]. In this section, we discuss their proposals and point out the differences with our approach. One of the important performance parameters in comparing caching of IP routing tables is the cache-hit/cache-miss ratio. We compare our ratio with each of these approaches knowing that these comparisons are inexact since the cache-hit ratio is dependent on the IP trace used and each of the proposals use different IP traces in their simulation.

4.1 Comparisons with Partridge study

This study was done on a backbone router using a 5-minute trace from FIX WEST. The total number of Internet packets was 4.97 million, which used 31.9 thousand total destination network addresses from the router [11,12]. We have compared it to two of our traces, which have both smaller and larger address ratios. Comparison results are shown in Table 9; blank spaces --- indicate that the data is not available.

Table 9. Comparisons of Cache-Hit Ratios

Cache Study	Partridge [11]	Prefix Memory Ternary Cache	
Trace Size	4.97M	3.27M	3.50M
Address Ratio*	156	102	229
Cache Size	*Cache Hit Ratio*		
1K	60.9%	70.8%	81.3%
2K	77.4%	82.2%	97.2%
4K	-------	93.2%	99.3%
5K	93.3%	-------	-------
8K	-------	97.6%	99.5%
10K	97.4%	-------	-------

**Address Ratio* is the number of trace packets divided by the number of unique destination network addresses. A high number may indicate a high temporal locality.

The cache-hit ratios from our scheme are higher, even for the trace (3.27M) that has lower Ratio, than those reported by BBN. Our scheme uses LRU cache replacement policy. In BBN experiments, none of the details of the cache were reported.

4.2 Comparisons with Chiueh and Pradhan study.

The paper by T. Chiueh and P. Pradhan [1] reports the results of a research effort on cache memory designs for Internet processors. This research included gathering of a large Internet trace file of 184.4 million packets. The two approaches are very different; ours uses variable-sized sets for caching the routing table prefixes, while Chiueh-Pradhan's employs a more conventional associative cache with set sizes of 1, 2, and 4 for Internet processors and caching IP addresses. As pointed out in [8] the number of possible IP routing prefixes is much smaller than the number of possible individual IP addresses (2^{32}) and therefore the effectiveness of caching IP routing prefixes had greater potential than caching IP addresses. We have compared to what seemed to be the best results of their designs, which were cache sizes of 4K and 8K entries with block size of one, and are called the host address range cache (HARC). We have shown the cache-miss ratio, which is defined as 100% minus the cache-hit ratio, in Table 10.

Table 10. Cache-Miss Ratio comparisons

		Prefix Memory Ternary Cache	Stony Brook
	Trace Size →	*12.5M*	*184.4M*
Cache Size	Associativity	Cache-Miss Ratio	
4K	1	------	7.54%
	2	------	4.58%
	4	------	3.64%
	Variable	1.72%	-------
8K	1	------	4.48%
	2	------	2.20%
	4	------	1.56%
	Variable	0.80%	------

The data shown under the Ternary Cache Study is the consolidation (12.5 million packets) of all six IP traces that were reported in this paper. The cache-miss ratio is the average of the six traces. This consolidation captures the effects of "flushing" the cache at the start of each of the six trace simulations.

The paper states that a port assignment would be made in one CPU cycle (which includes one cache memory cycle) if the assignment is found in cache. This compares with our port assignment in just one cache memory cycle when the assignment is in cache. The caching of IP addresses does not add any port errors to the performance. From Table 10, it can be observed that our scheme leads to cache-miss ratios of about half of the best performance on the HARC scheme. This study proposes dealing with cache coherency problems by flushing the cache whenever the routing table receives a change notice. Since these changes occur frequently that approach would be very stressful on the hardware. Our approach would be to use the sampling technique to minimize cache coherency and change the cache less frequently by flushing or other means

4.3 Comparisons with Liu study.

The paper by H. Liu [8] reports the simulation results of a scheme for routing table prefix caching in a network processor design using a TCAM as the cache memory. This scheme has some similarities to our approach in that we both propose routing prefix caching in a TCAM bases structure. The major focus of this paper is to eliminate the potential of passing from the routing table to the cache, entries that are not the longest prefix match, while adding the performance enhancement of a TCAM cache. In the routing table, the longest prefix match is defined as the superprefix while other matches are defined as subprefix. Three designs are presented:

1) *Complete Prefix Tree Expansion* (CPTE) transforms the routing table into a complete prefix tree so that no IP lookup will match a subprefix and thereby cause a potential port error. This approach required the expansion of the routing table between 45% and 118% for the traces that were studied.

2) *No Prefix Expansion* (NPE) identifies all subprefixes in the routing table and marks them as non-cacheable. When a non-cacheable entry is a match in the routing table the complete IP address and a complete mask are returned to cache. This does not require expansion on the routing table, but one must keep track of non-cacheable routing table entries and the cache becomes a hybrid between an IP prefix cache and an IP address cache.

3) *Partial Prefix Tree Expansion* (PPTE) only expands the routing table partially (13% to 20% in the traces studied), limiting the expansion to first level nodes. This reduces the number of subprefixes that an IP lookup could match but those remaining must be marked as non-cacheable and returns the total IP address and mask if they become a routing table match. Again the cache becomes a hybrid between a routing table prefix cache and an IP address cache.

A cache-miss ratio comparison with this approach is shown in Tables 11 and 12. The tables compare our proposal (Prefix Memory Ternary Cache) with the three proposed schemes and an additional simulation called IP (only). The traces used in the two studies are completely different but representative of IP traffic. The IP (only) cache simulation results were provided in [8] for comparisons, it uses the same TCAM cache approach but caches IP address only. Huan Liu shows simulations using three different IP traces. We compare the results of two of these traces to our results from similar traces. We did not have a single trace as large at the 12.8M packet trace to compare with.

Table 11. Cache-miss ratio comparisons

Cache Study	Huan Liu				Prefix Memory Ternary Cache	
Trace Size	1.86M (SDC 958)				1.77M BWY-1001908207-1	1.82M TAU-1007728734
Address Ratio*	134				312	189
	Cache Miss Ratio					
Cache Size	CPTE	PPTE	NPE	IP (only)		
8K	0.26%	0.32%	0.40%	0.75%	0.33%	0.51%
4K	0.27%	0.34%	0.50%	0.95%	0.35%	0.65%
2K	0.36%	0.50%	0.70%	1.6%	0.56%	1.24%

Table 12. Cache-miss ratio comparisons

Cache Study	Huan Liu				Prefix Memory Ternary Cache
Trace Size	0.63M (970222)				0.20M ADV-1007598319-1
Address Ratio*	62				139
	Cache Miss Ratio				
Cache Size	CPTE	PPTE	NPE	IP (only)	
8K	0.57%	0.66%	0.80%	1.60%	0.80%
4K	0.57%	0.66%	0.80%	2.40%	0.80%
2K	0.88%	1.10%	1.50%	3.7%	0.80%

**Address Ratio* is the number of trace packets divided by the number of unique destination network addresses. A high number may indicate a high temporal locality.

We should expect similar results between our approach and the CPTE approach since they are both pure IP prefixes caches using conventional TCAM architectures. The differences shown in Tables 11 and 12 may be attributed to the use of different IP traces and the fact that:

a) Some destination IP addresses in Liu's simulations are omitted [8].
b) Our decoding of the sanitized IP addresses did not capture geographical locality.

To address cache coherency concerns, Liu's approach adds an invalidation command that cancels any routing entry in the cache that is no longer valid. This in turn not only adds complexity to the system but also affects performance. Below are the steps that may need to be added to handle a new routing prefix.

1. Check in routing table if entry is cached.
2. Send an invalidation command.
3. Compute prefix tree expansion of affected entries.
4. Replace entries in cache.

In addition, any time that an entry is deleted from cache, this deletion needs to be communicated to the routing table.

4.4 Summary of Comparisons.

Our approach is different from all previous approaches for caching routing table look-ups. Chieuh and Pradhan [1] propose an IP address cache embedded in a conventional Internet processor to cache IP addresses. Liu's [8] proposal is similar to ours in that it uses a TCAM based prefix cache for caching routing table prefixes and masks but requires a transformation of the routing table into a complete prefix tree. This transformation resolves the problem of mismatches that could cause the wrong port selection

but result in significant increase in routing table size and complexity. Our approach is to use a sampling technique to reduce the mismatches to a manageable problem without requiring expansion to the routing table. To resolve cache coherency concerns, Chieuh and Pradhan [1] propose flushing the cache every time the routing table is changed, which is stressful on the hardware while Liu [8] proposes the addition of an invalidation command that cancels any routing entry in the cache that is no longer valid. This addition adds complexity to the system and affects performance. Our proposed sampling technique can be used to minimize the problem until the cache can be updated on a maintainable schedule whether by flushing the cache or other methods.

5. Concluding Remarks

In this paper we have presented a study of an associative cache scheme for Internet IP routing. This is the first study that deals with variable set associative caches for this application. In our approach an output port assignment requires only one cache memory access when the assignment is found in cache. When it is not found in cache, the routing tables must be accessed.

Our study shows that an associative cache provides an output port at the speed of one memory access with a very high hit rate. A list of the main features of the proposed associative ternary cache is provided below.

- *Sampling technique to reduce port misses.* A sampling technique has been introduced to significantly reduce port misses. It has been shown that the worst-case port error rate, caused by not having the longest prefix match in cache, could be reduced by an order of magnitude (from 0.52% to just 0.08%). The sampling technique also reduces port misses caused by cache incoherency due to routing table updates. Sampling is an alternative to flushing the cache to assure cache coherency every time the routing table is updated. It was also shown that the number of cache writes is significantly reduced as the cache size was increased from 1K to 8K entries; often as much as an order in magnitude. The cache writes due to our sampling technique adds at most 0.65% writes to the cache.
- *High cache hit rates.* A high cache hit rate is achieved with relatively small cache sizes; this is due to the high temporal locality exhibited by these IP trace files. The 4K and 8K-entry cache hit rates ranges from 93.21% to 99.65% and from 97.62% to 99.67%, respectively.
- *LRU replacement policy.* The least recently used (LRU) replacement policy has shown to yield lower cache-miss ratios than other two policies (random and FIFO) by as much as 37%.
- *Comparison with other scheme.* Our scheme, when compared with two other approaches [1, 11], shows a better cache hit ratio, which is the primary measure of performance for cache schemes. When our approached is compared to Liu's scheme [8], it shows comparable cache miss rate without modifications to the routing table. Liu's approach does not introduce port errors.

Our approach imposes no additional delays to IP routing schemes since the same address that is sent to the routing tables is used for the cache. The cache entries are inserted as the routing table provides the port assignment to a given address. The sampling technique updates the cache in much the same way, inserting the correct routing table entry into cache after the cache-hit selection has been forwarded. Thus, the cache scheme is completely transparent and improves the performance of the router node.

The proposed associative cache IP router scheme introduces a novel way to deal with sets, which are traditionally of a fixed size. In our scheme we allow some sets to be much larger than others to accommodate the statistics of routing table prefix distributions.

Acknowledgements

This work was supported in part by the National Science Foundation under contract CCR9900643. The authors would like to thank the anonymous reviewers of this paper for their constructive comments and suggestions.

References

[1] T. Chiueh and P. Pradhan, "Cache Memory Design for Internet Processors," *6th Symposium on High Performance Computer Architecture (HPCA-6)*, Toulouse, France, January 2000.

[2] J. G. Delgado-Frias and J. Nyathi, "A High-Performance Encoder with Priority Lookahead," *IEEE Transactions on Circuits and Systems, I: Fundamental Theory and Applications*, vol. 47, no. 9, pp. 1390-1393, September 2000.

[3] W. Doeringer, G. Karjoth, M.Nassehi, "Routing on Longest-Matching Prefixes," *IEEE/ACM Transactions on Networking*, Vol. 4, No. 1, pp. 86-97, February 1996.

[4] P. Gupta, S. Lin, and N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds," *IEEE InfoCom*, San Francisco, April 1998.

[5] N. Huan, W. Chen, J. Luo, and J. Chen, "Design of multi-field IPv6 packet classifiers using ternary CAMs**,"** *IEEE Global Telecommunications Conference, 2001. GLOBECOM '01*, vol. 3, pp. 1877-1881, 25-29 Nov. 2001.

[6] V. Lines, A. Ahmed, P. Ma, S. Ma, R. McKenzie, H. Kim, and C. Mar, "66 MHz 2.3 M ternary dynamic content addressable memory," *2000 IEEE International Workshop on Memory Technology, Design and Testing*, pp. 101-105, 7-8 Aug. 2000.

[7] H. Liu, "Routing table compaction in ternary CAM," *IEEE Micro*, vol. 22, no. 1, pp. 58-64, Jan.-Feb. 2002.

[8] H. Liu, "Routing Prefix Caching in Network Processor Design," *Proc. International Conference on Computer Communications and networks (ICCCN)*, Phoenix, AZ, 2001.

[9] Merit Networks Inc., *Internet Performance Measurement and Analysis (IPMA) project*, a joint effort of the University of Michigan Department of Electrical Engineering and Computer Science and Merit Network. The web site is: http://www.merit.edu/ipma.

[10] National Laboratory for Applied Network Research, *Passive Measurement and Analysis Project of the National Laboratory for Applied Network Research at San Diego Super Computer Center*. The web site is: http://pma.nlanr.net/PMA.

[11] C. Partridge, "Locality and Route Caches," Position Statement for the *1996 NSF Workshop on Internet Statistics Measurement and Analysis*. Available at: http://www.caida.org/outreach/isma

[12] C. Partridge et. al., "A 50-Gb/s IP Router," *IEEE/ACM Transactions on Networking*, vol.6, no.3, June, 1998.

[13] Y. Rekhter and T. Li, "An Architecture for IP Address Allocation with CIDR," *RFC 1528*, September 1993.

[14] D. H. Summerville, J. G. Delgado-Frias, and S. Vassiliadis, "A Flexible Bit-Associative Router for Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 5, pp. 477-485, May 1996.