International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

A Survey on Different IP Address Lookup Approaches

Manohar Nelli V

Department of Computer Science and Engineering, Jawaharlal Nehru National College of Engineering, Shivamogga, India
manohar[at]jnnce. ac. in

Abstract: ***Due to the rapid growth of traffic in the Internet, backbone links of several Gigabit/sec are commonly deployed. To handle Gigabit/sec traffic rates, the backbone routers must be able to forward millions of packets per second on each of their ports. Fast IP address lookup in the routers, which uses the packets destination address to determine for each packet the next hop, is therefore crucial to achieve the packet forwarding rates required. IP address lookup is difficult because it requires a longest matching prefix search. In the last couple of years, various algorithms for high performance IP address lookup have been proposed. A survey of IP address lookup algorithms is presented here.***

Keywords: Host address caching, Multibit, GPU, CUDA

1. Introduction

The primary role of routers is to forward packets towards their final destination. To this purpose, a router must decide for each incoming packet where to send it next. More exactly, the forwarding decision consists in finding the address of the next - hop router as well as the egress port through which the packet should be sent. This forwarding information is stored in a forwarding table that the router computes based on the information gathered by routing protocols. To consult the forwarding table, the router uses the packet's destination address as a key; this operation is called address lookup. Once the forwarding information is retrieved, the router can transfer the packet from the incoming link to the appropriate outgoing link, in a process called switching. The exponential growth of the Internet has stressed its routing system. While the data rates of links have kept pace with the increasing traffic, it has been difficult for the packet processing capacity of routers to keep up with these increased data rates. Specifically, the address lookup operation is a major bottleneck in the forwarding performance of today's routers. This paper presents a survey of the latest algorithms for efficient IP address lookup. Here, various approaches to lookup IP addresses efficiently are presentedwhich helps to understand the address lookup problem and it also provides a suitable solution for every problem which is occurring due heavy and explosive internet traffic that has given rise to many problems.

2. High - Performance IP Routing Table Lookup Using CPU Caching

Tzi - cker Chiueh and Prashant Pradhan [1] proposed Suez project to demonstrate that general - purpose CPU can serve as a powerful platform for high - performance IP routing. Suez's routing table lookup algorithm is based on two data structures; a destination host address cache (HAC), and a destination network address routing table (NART). Both are designed to use CPU cache efficiently. The algorithm first looks up the HAC to check whether the given IP address is cached in the HAC because it has been seen recently. If so, the lookup succeeds and the corresponding output port is used to route the associated packet. If not, the algorithm further consults the NART to complete the lookup.

a) Host Address Caching

Typically multiple packets are transferred during a network connection's lifetime, and the network route a connection takes is relatively stable. Therefore the destination IP address stream seen by a router exhibits temporal locality. That is, the majority of the routing table lookups are serviced directly from the HAC. Data flow of HAC is as shown in fig 1.

Minimizing the HAC hit access time is crucial to the overall routing table lookup performance. Rather than using a software data structure such as a hash table, Suez's HAC is architected to be resident in the Level - 1 (Ll) cache at all time, and to be able to exploit the cache hard - ware's lookup capability directly. As a first cut, 32 - bit IP addresses can be considered as 32 - bit virtual memory addresses and simply looked up in the L1 cache. Suez's HAC lookup algorithm takes a combined software and hardware approach. To reduce virtual address space consumption, Suez only uses a certain portion of each IP address to form a virtual address, and leaves the remaining bits of the IP address as tags to be compared by soft - ware. This approach makes it possible to restrict the number of virtual pages reserved for the HAC to a small number.

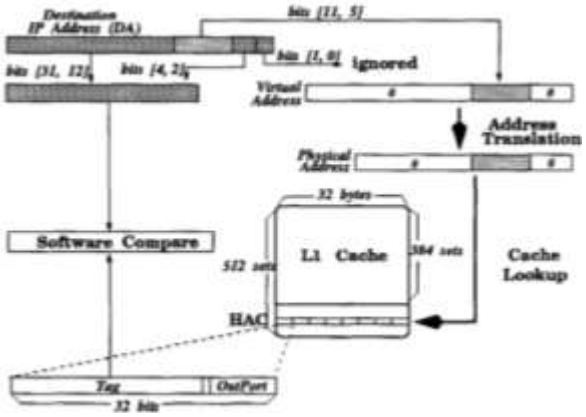

Figure 1: Data flow of HAC

If they hit in the L1 cache, the lookup is completed in one CPU cycle; otherwise an NART lookup is required.

b) Network Address Routing Table

If the HAC access results in a miss, a full - scale NART lookup is required. The guiding principle of Suez's NART design is to trade table size for lookup performance. This principle is rooted in the observation that the L2 cache size of modern microprocessors is comparatively large for storing routing tables and the associated search data structures, and is expected to continue increasing over time. The same above said approach as in case of HAC (host address caching) is implemented for NART (network address routing table).

The main advantage is that, it speeds up the lookup process of an IP address by avoiding visiting unrelated routing table entries. By treating IP addresses as virtual memory addresses, one can exploit CPU caching as a hardware assist to speed up routing table lookup significantly. Suez's routing table lookup algorithm is simple and fast and does not require backtracking to support longest prefix match. This makes easy to traverse Suez's routing table at an increased speed which will serve in forwarding pockets at a faster rate.

Disadvantage is that, more complex machinery is required to twist the CPU cache as a routing table lookup cache, because the "tags" of IP addresses are of variable length, whereas existing CPU cache hardware only supports fixed - length tags. There is much less spatial locality in the destination host address stream compared to the memory reference stream in typical program execution.

3. An Improvement of IP Address Lookup based on Rule Filter Analysis

Perez, K. Guerra, Xin Yang, and SakirSezer [16] proposed a rule filter analysis to improve IP address lookup. A rule is composed of five or more fields and it defines an action. When an input packet matches against a rule, the corresponding action is applied to the input packet. A set of determined rules is called a filter. Consequently, different kinds of filters are examined: Accesses Control List (ACL), Firewall (FW) and IP Chain (IPC), with different sizes. The size of the rule filters is named as 1 K, 5 K and 10 K rules in order to simplify the denomination of rule sets.

a) Original Multi - bit Search Trie

Each node of the original Multi - bit Search Trie represents a determined n - bits prefix in the trie algorithm. Each leaf node stores a list of rules and the highest priority matching rule (HPMR) is found using a simple linear search. Using this methodology, it is expected that memory space as well as long lookup time will be inefficient due to the list of rules stored in each trie node. However, supposing there are no repeated rules, this experiment runs at a fast insertion process. Different scenarios are studied for IPv4 using tries with four levels per dimension, in order to acquire the optimal parameters values.

b) Multi - bit Trie with labelled rule fields Experiment 1 (EXP_1)

Experiment 1 (EXP_1) is based on an improved structure of the original Multi - bit Search trie algorithm. According to the rule filter analysis, EXP_1 performs the lookup process using the label method. It demonstrates that the number of unique rules is lower than the total number of rules. Thus, the label represents all rules containing this field. The main idea of this work is to label each unique rule field. By storing the labels instead of the entire rule information, memory consumption can be significantly reduced. Here, a label is assigned to the unique 16 bit partitions of each rule field that must be stored in the multi - bit tries. Consequently, each trie links with a certain label filter. The independent filter information is composed of a label. This will require less memory storage than original Multi - bit trie.

With this method, we expect that this experiment will require less memory storage than original Multi - bit trie. Furthermore, the lookup process is expected to be faster. However, the update processes can be compromised by the corresponding label lookup into the filters.

c) Multi - bit Trie with labelled nodes Experiment 2 (EXP_2)

Experiment 2 (EXP_2) uses label method on a multi - bit trie. In this case, the trie nodes are labelled instead of the unique field. After all search results are available from each trie, the final lookup is performed in another label filter with combinations of labels. The experiment demonstrates not only a reduction of memory space, but also an improved lookup speed. Since leaf nodes do not contain any rule list, the goal of EXP_2 is also to avoid the linear search into the trie. Moreover, the corresponding label does not have to be searched through a filter beforehand. The label will be retrieved when the leaf node is reached.

Advantage of this approach is that, multi - bit tries still do linear search on lengths, but since the trie is traversed in larger strides the search is faster. This method reduces the depth of the trie and it is an easy hardware solution mapped into pipeline stages. It reduces the update time in comparison with other trie structures.
Limitation is the requirement of intensive design tasks on time/space complexity, a very large number of rules, high speed, scalability, flexibility. This method requires higher latency and more storage with a larger address width. There is a need to store children nodes for each new created node, denoting an inefficient memory usage. This is not suitable for a large number of entries and it does not support incremental update.

4. An On - Chip IP Address Lookup Algorithm

Sun, Xuehong, and Yiqiang Q. Zhao [3]proposes a new data compression algorithm to store the routing table in a tree structure using very little memory. The data structure used in this method is tailored to a hardware design reference model presented here.

a) Hardware design reference model

The IP destination address enters the chip as a key for looking up the next hop information. The output of the chip is an index to the external SRAM where the port information can be found. The chip is an ASIC that consists of a memory system and an ALU part. The memory system uses on - chip SRAM. Using an IBM blue logic Cu - 08 ASIC process, the I/O width of the SRAM to the control logic unit can be as wide as 144 bits. The memory access time can be as low as 1.25ns. The size of the on - chip memory is about 1 Mbits. Assuming there are 144 bits in each row, the chip has less than 213 rows altogether. This means 13 bits is enough to index into any row of the memory. The ALU receives keys from outside and produces outputs to the outside. It may access the memory of the system and it performs some simple logic and arithmetic operations. According to different design goals, the chip can be configured or programmed. In fact, this reference model shown in fig 2 can be modified to be tailored to different situations.

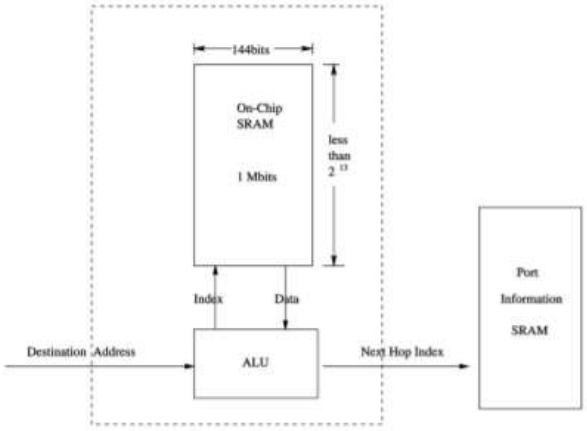

Figure 2: Hardware design reference model

The reason to choose SRAM instead of DRAM as the on - chip memory is that the SRAM access time is several times faster than the DRAM. The on - chip SRAM memory size can be made more than 100M bits large, which is well suitable for any IP lookup task.

Advantage is that, the on - chip memory access latency is very low. It provides larger bus width to on - chip memory than that to off - chip memory. The number of pins of the chip is smaller if the memory goes on chip rather than off chip. The algorithm can be implemented in one single chip. Updating process will not interfere with the searching.

Disadvantage is that, the memory cannot be large, only suitable for smaller memory. Memory requirement increases proportionally to the increase of the number of prefixes. Due to the lack of techniques to compress and optimize the data structures, number of memory accesses is more.

5. IP Address Lookup by Using GPU

Hung - mao chu, Tsung - hsien li, Pi - chung wang [4] proposes a system which uses a CUDA based architecture which has an inherent hazard that one copy of all data and at least one device function must be acquired from the main memory of the computer system via PCIe bus. When the computing task is accomplished, the results are duplicated and transmitted reversely. When the data must be retrieved from other components like NICs, the PCIe bus becomes a bottleneck for transmitting the data for CUDA computing. In order to solve this problem, GPUDirect remote direct memory access (RDMA) technology is contrived to enable third party endpoints directly communicate each other. CUDA GPUs implement a base address register (BAR) which allows devices directly access the internal memory of a GPU device without passing through system memory or CPU.

a) Proposed Architecture

The conceptual architecture of the proposed GPU based IP forwarding engine is shown in fig 3, where the GPU is only responsible for performing IP address lookups. In this architecture, NICs directly transfer headers of packets to the global memory of GPU by the fast on - board memory bus. Although transmitting IP addresses via a PCIe bus is possible, the bus bandwidth (8 GB/s per direction with 16 lanes) is not sufficient, as demonstrated. The CPU constructs and transmits the data structure to the global memory via relative slow PCIe bus. By instructing CUDA compiler, the searchable data structure is stored in the texture memory to benefit from the texture cache and shortens the access latency. Both packet headers and the searchable data structure are accessed through the CUDA memory bus.

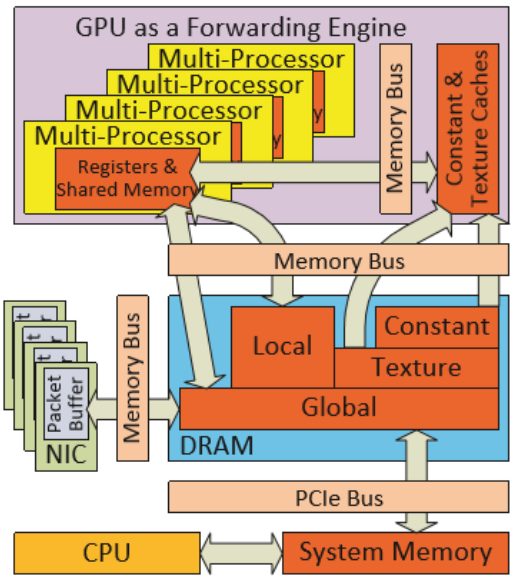

Figure 3: GPU based IP forwarding engine

The number of threads which execute the IP address lookup procedures simultaneously according to the packet buffer size is determined. For instance, each gigabit Ethernet (GbE) port is equipped with 42 MB buffer for Juniper L3 switch. The buffer is increased to 512 MB for each 10 GbE port. Cisco XR 12000 series routers are equipped with 512MBforbothone - port10GbEand10 - portGbE linecards. These line cards use buffers to transmit and receive packets. According to these specifications, more than 350K packets can be stored in a router per 10 Gbps throughput. Considering the number of packets is significantly higher than the number of allowable GPU threads, here the number of threads is maximized in the experiments. The packet latency may be increased because of the long lists of packets

waiting for the results of IP address lookups. However, experimental results show that a GPU can achieve sufficient throughput to support more than one thousand GbE ports. Thus, the packet latency can be reduced significantly to make the configuration feasible. The packet path in the proposed architecture in the data plane and the control plane is described. The path of data plane is listed below. The incoming packets received by a NIC are stored in the buffer. A batch of packet headers is transmitted to global memory. Streaming multiprocessors (SMs) read the headers from global where each header is assigned to one thread. SMs search the data structure in the texture cache/memory to determine the next - hops of all headers. The lookup results for each batch are stored in another pre - assigned address of the global memory. The lookup results in the global memory are pushed back to the corresponding NICs to accomplish packet forwarding.

Besides packet forwarding, routers also need to provide control plane and management plane functions. For example, a router needs to ensure that the contents of the forwarding table reflect the current network topology by receiving route update messages from neighbouring routers. When the NICs receive packets for control or management plane, these packets are forwarded to the CPU for further processing. The CPU may also transmit control messages to other routers. These messages are transmitted to one or more NICs through theglobal memory of CUDA. The packet path of the control messages is listed below. NICs identify that the received packets are control messages. The control packets in NICs are transferred to a designated region in the global memory. The control packets in the global memory are moved to the system memory through PCIe bus. The CPU processes control packets and generates response packets in the system memory. The response packets are delivered to the global memory via PCIe bus and transmitted to the packet buffer of NICs for forwarding as a general data packet.

The next - hops of the control packets are determined by the CPU itself to avoid interfering the procedure of data plane. Since only a small percentage of packets are control packets, the lookup overhead of CPU and the bandwidth consumption of both PCIe bus and global memory are acceptable.

6. Conclusion

This paper describes different approaches used for IP address lookup at a faster rate. High - performance software - based IP routing table lookup algorithm exploits CPU caching aggressively. The algorithm uses a portion of the CPU cache to support destination host address caching. Another approach uses rule filter analysis to lookup IP addresses and based on the specified rule, packets are classified and then forwarded. On - chip algorithm has a very small memory requirement. With this advantage, routing table can be put into a single chip, thus, the memory latency to access the routing table can be reduced which improves the performance. Next approach for IP lookup is to make use of CUDA, a programmable GPU which reduces overhead on CPU and GPU memory has sufficient bandwidth to fully utilize the computation capability of processor to forward the packet with reduced latency.

References

[1] Chiueh, Tzi - cker, and Prashant Pradhan. "High - performance IP routing table lookup using CPU caching. " INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol.3. IEEE, 1999.
[2] Perez, K. Guerra, Xin Yang, and SakirSezer. "An improvement of IP address lookup based on rule filter analysis. " 2014 IEEE International Conference on Communications Workshops (ICC). IEEE, 2014.
[3] Sun, Xuehong, and Yiqiang Q. Zhao. "An on - chip IP address lookup algorithm. " IEEE Transactions on Computers 54.7 (2005): 873 - 885.
[4] T. H. Li, H. M. Chu, P. C. Wang, "IP address lookup using GPU", Proc. IEEE 14th Int. Conf. HPSR, pp.177 - 184, Jul.2013.