A Memory Efficient, 100 Gb/sec MAC Classification Engine

V. Papaefstathiou, I. Papaefstathiou
Foundation of Research & Technology Hellas (FORTH),
Institute of Computer Science (ICS),
Vassilika Vouton, GR71110, Heraklio, Crete, Greece
{papaef, ygp}@ics.forth.gr

Abstract

In this paper, we propose a classification engine employed at Ethernet's MAC Layer which uses an innovative hashing scheme and internal replacement of MAC Vendor IDs; the *Hash Based Classification Engine (HBCE)* compacts the MAC address tables and supports extremely high speed decisions. Its very low memory requirements make it a very promising candidate for Ethernet equipments that would need to support classification at Data Link Layer and at Multi-Gigabit per second network speeds. Moreover HBCE can also so be used very efficiently in lower-bandwidth wireless environments.

1 Introduction

Ethernet is, by far, the most common Layer-2 network protocol. Due to, mainly its high performance-cost ratio, it is currently making a breakthrough in MANs and WANs, as well as in wireless networks. The deployment of Multi-Gigabit wired or Lower-bandwidth wireless Ethernet networks, and their use beyond the tight borders of LANs, motivated the development of QoS mechanisms in the MAC layer, such as the VLAN scheme, or certain QoS protocols for wireless environments [1]. Those mechanisms require identification of network flows and classification of Ethernet packets according the following three MAC-header's fields:

1. MAC Addresses : 48-bits
2. VLANs: ID is 12-bits
3. Ports (equipment specific)

The length of the MAC addresses, namely 48-bits, is what makes those decisions difficult since exact matches in such a wide value are not a trivial task. As a result many implementations use expensive and power consuming CAMs or trie based solutions that have poor overall performance. Another popular approach is hashing of the MAC address bits in order to generate an index for a lookup table. Many of these solutions use the CRC polynomials for hashing since they have been proved very efficient [2] or others use direct mapping.

In this paper, we propose a classification engine for the MAC layer of the Ethernet networks which uses a new hashing scheme and internal replacement of MAC Vendor IDs; the *Hash Based Classification Engine (HBCE)* compacts the MAC address tables and supports high speed decisions using significantly less memory than the existing solutions.

2 Hash Based Classification Engine

HBCE is based on popular hashing, but we propose a hashing scheme that exactly matches the special characteristics of the MAC addresses while it is designed to support tens-of-thousands of MAC-address rules and a couple of thousands of VLAN-based and port-based rules. HBCE focuses on the MAC address lookup scheme since it is the most critical part in terms of both speed and storage. VLANs and ports are relatively small in size and are directly mapped into tables.

2.1 MAC classification scheme

Studying the published IEEE OUI and Company ID Assignments we have realised that the 24-bit vendor address space of the MAC addresses is not fully occupied. In fact, fewer than 8000 vendors are active instead of the 2^{24} possible. Therefore we can replace the 24-bit vendor ID with a 13-bit internally assigned vendor ID; the last 24-bits of the MAC address remain unchanged. This internal replacement of the vendor IDs can help us reduce the storage requirements, at the cost obviously of the replacement operation. This replacement means that we keep a small decoding table with 8192 entries that assigns an internal 13-bit value for every 24-bit Vendor ID.

After this replacement we define a hashing function on the resulted 37-bits of the MAC address. The MAC addresses are stored in a 64K table called MAC_TBL and the indexes to it are generated by our hashing function. The collisions due to hashing are handled by pointers to variable size blocks with dynamic memory management implementation [3].

The 16-bit indexes in MAC_TBL are generated as follows:

$$MAC\_TBL_{index} = \{MAC[31:24] \ \textbf{xor} \ MAC[15:8] \ , \ MAC[23:16] \ \textbf{xor} \ MAC[7:0] \}$$

To identify a certain MAC address we save some additional information so as to distinguish the MACs that collide. The advantage of our scheme is triggered by the fact that we don't need to save all 48-bits; a MAC located in address A of MAC_TBL can be reproduced by the 16-bits of A and a quantity (H_{val}) of the MAC address. The H_{val} is 21-bits and is defined as follows:

$$H_{val} = \{ MAC[36:24] \ , \ MAC[7:0] \}$$

Then the MAC addresses can be reproduced by the 16-bits of the address A and H_{val} as follows:

$$MAC[36:24] = H_{val}[20:8]$$
$$MAC[23:16] = A[15:8] \ \textbf{xor} \ H_{val}[7:0]$$
$$MAC[15:8] = A[15:8] \ \textbf{xor} \ H_{val}[15:8]$$
$$MAC[7:0] = H_{val}(7:0)$$

2.2 Data Structures

In order to resolve the possible collisions in MAC_TBL we define a complex data structure. Each MAC address stored in an entry of the table needs 21-bits (H_{val}) to be fully identified and 15-bits for the corresponding Flow Identifier (FlowID); support of 32K distinct network flows is sufficient for the majority of the target Ethernet devices. In the case where only one MAC address is saved in a table entry we can save the FlowID in the 15 MSB of the word and H_{val} in the 21 LSB. Empty table entries (not mapped to any MAC) are indicated by FlowID number 0 and entries mapped to many MAC addresses are indicated by Flow ID number 1. In the latter case, where collisions occur, we also store a pointer to the variable size block and the number of MACs that collide. The number of colliding MACs can also indicates the size of the linked block. The last 17-bits of the word are used to store the pointer to the block and the remaining 4-bits are used to keep the number of collisions; these 4-bits have been found enough for the maximum number of collisions of our system.

3 Performance and Implementation

In order to measure the performance of our scheme, we have generated databases containing 32K and 64K MAC-address based rules, with variable number of active vendor IDs, such as 1500 and 4000. For the generation of the databases we used real MAC Vendor IDs from the subset provided by OUI and appended random uniformly distributed 24-bit values that can represent the real network cards' serial numbers.

In Figure 1 we demonstrate results of the average number of collisions of the proposed hashing scheme and compare it with CRC-16 and direct mapping by the 16 LSBs. We can se that the HBCE seems a very effective hash function that has similar performance to CRC-16, while it is more efficient than direct mapping. The main advantage of HBCE, when compared with CRC, is that it requires only a small portion from the original MAC address to be stored instead of the complete required by CRC; CRC does not have an inverse function. The results also show that the number of active vendors in the dataset seems not to influence the performance significantly.

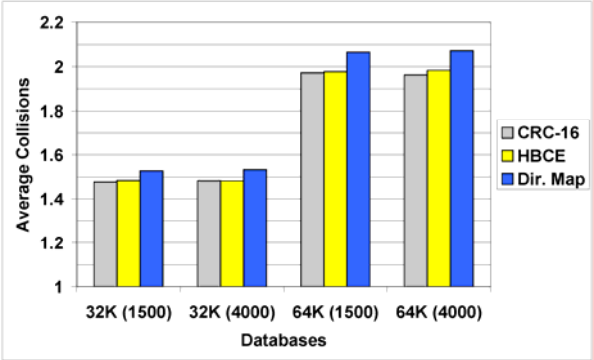

Figure 1. Average Observed Collisions

In Table 1 we present the storage requirements and the network performance of HBCE based on the generated databases. We can see that half megabyte is enough for HBCE to store rules corresponding to 64K MAC addresses. More importantly, HBCE has 32% - 41% lower storage requirements than the schemes based on the CRC-16 hashing function. Note that here we assume 36-bit wide SRAM memory words.

Table 1. HBCE storage and network performance

DB Size (Active Vendors)	CRC RPL Total (KB)	CRC Total (KB)	HBCE Total (KB)	Average Throughput 400Mhz (Gbps)
32K (1500)	626	590	395	138,1
32K (4000)	**626**	**590**	**396**	**138,3**
64K (1500)	940	904	532	103,9
64K (4000)	**938**	**902**	**533**	**103,5**

We have implemented a fully pipelined hardware realization of HBCE that works in frequencies beyond 450Mhz in 0.13μm ASICs and 100Mhz in FPGAs, utilizing less than $0.1mm^2$ of silicon area. This implementation requires less that 5 clock cycles on average for a classification decision of a 64K database and can handle traffic volumes higher than 100 Gbit/sec of minimum sized 64-byte Ethernet packets.

4 References

[1] Giovanni Pau et.al , *"A Cross-Layer Framework for Wireless LAN QoS Support"*, IEEE ITRE, August 11-13, 2003, Newark, New Jersey, USA.

[2] R. Jain, *"A Comparison of Hashing Schemes for Address Lookup in Computer Networks"*, IEEE Transactions on Communications, Vol. 40, No. 3, October 1992, pp. 1570-1573.

[3] V.Papaefstathiou, *"Design & Implementation of Network Packet Classification Engines"*, M.Sc. Thesis, CSD, University of Crete, Greece, 2005.