

# Algorithmic Approaches to Redesigning TCAM-Based Systems

Chad R.Meiners Alex X. Liu Eric Torng  
 Department of Computer Science and Engineering  
 Michigan State University  
 East Lansing, MI 48824, U.S.A.  
 {meinersc,alexliu,torgng}@cse.msu.edu

**Categories and Subject Descriptors:** C.2.6 [Internet-working]: Routers C.2.1 [Network Architecture and Design]: Network communications; Packet-switching networks

**General Terms:** Algorithms, Performance, Security.

**Keywords:** Packet Classification, TCAM, Range Expansion, Pipeline.

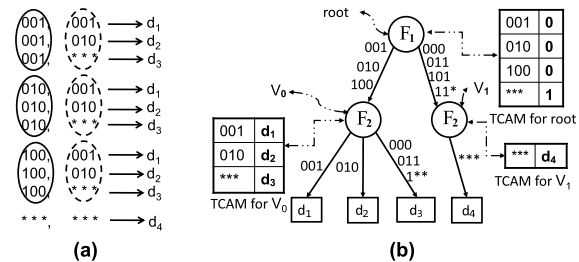
## 1. INTRODUCTION

Using Ternary Content Addressable Memories (TCAMs) to perform high-speed packet classification has become the *de facto* standard in industry because TCAMs enable constant time classification by comparing a packet with all rules of ternary encoding in parallel. However, TCAMs have limitations of small capacity, large power consumption and heat generation, and high hardware cost. Although a hardware solution to TCAM limitations is not impossible, TCAMs are unlikely to have hardware breakthroughs because they have pushed silicon to its limit. Furthermore, the number of rules in packet classifiers increases rapidly due to the explosive growth of services deployed on the Internet.

In this paper, we propose three approaches, *multi-lookup*, *pipelined-lookup*, and *packing*. The central theme of these three approaches is to minimize the number of TCAM bits used to represent a packet classifier. Reducing TCAM space usage directly addresses the physical limitations of TCAMs. Smaller TCAM implies lower power consumption, less heat generation, less board space, and lower hardware cost. Furthermore, reducing the number of bits used in a TCAM leads to less power consumption and heat generation because the energy consumed by a TCAM grows linearly with the number of bits it uses in storing rules.

Our approaches are based on three key observations. First, information stored in TCAMs tends to have high redundancy from an information theory perspective. Specifically, we observe that the same ternary string for a specific field may be repetitively stored in multiple TCAM entries. For example, in the simple two-dimensional packet classifier in Figure 1(a), the strings 001, 010, and 100 from the first

field are each stored three times in the TCAM. Second, a TCAM search key can be ternary. A TCAM chip typically has a built-in Global Mask Register (GMR) that supports ternary search keys. The GMR of a TCAM chip contains a bit mask that specifies which bit columns in the chip participate in a search. A GMR allows multiple lookup tables to be packed into one TCAM chip where the GMR can be used at run time to dynamically select the right table to search. Third, in current TCAM-based packet classification systems, the TCAM width exceeds the TCAM bus width; the result is that one search requires several cycles. TCAM chips typically can be configured to be 36, 72, 144, or 288 bits wide. As the width of the 5-tuple (source IP, destination IP, source port, destination port, protocol type) is 104 bits, TCAM chips are typically configured to be 144 bits wide. However, the TCAM bus width is typically 72 bits. Therefore, the throughput of traditional single-lookup approaches is only one packet per five cycles.



ket  $(p_1, \dots, p_d)$  can be found by  $d$  searches on the TCAM. The first search key  $k_1$  is formed by concatenating the root node's ID and  $p_1$ . Let  $f(k_1)$  denote the search result of  $k_1$ . The second search key,  $k_2$  is formed by concatenating  $f(k_1)$  and  $p_2$ . This process continues until we compute  $f(k_d)$ , which is the decision for the packet.

### 3. PIPELINED-LOOKUP APPROACH

The multi-lookup approach is an effective method for reducing TCAM space needed for packet classifiers. However, this reduction in space reduces packet classification throughput by requiring multiple lookups on a single TCAM chip. Our pipelined-lookup approach improves throughput by using one TCAM chip for each field. That is, we will use five TCAM chips where, for  $1 \leq i \leq 5$ , chip  $i$  stores table  $t_i$  which is the merger of all tables of  $F_i$  nodes. Having one merged table per field in a separate TCAM chip enables us to pipeline the multiple lookups needed for processing each packet. Because the tables generated by the pipelined-lookup approach are thinner than the TCAM bus width, the pipelined-lookup approach can be four to five times faster than the traditional single-lookup approach. Furthermore, separating the tables from different fields yields new opportunities to save bits.

In the pipelined-lookup approach, a  $d$ -dimensional packet search is separated into  $d$  searches, which are further pipelined. That is, the  $d$  TCAM chips are chained together into a pipeline such that the search result of the  $i$ -th chip is part of the search key for the  $(i + 1)$ -st chip, and the result of the last chip is the decision for the packet. With such a chain,  $d$  packets can be processed in parallel in the pipeline. Figure 2 illustrates the pipelined-lookup process of packet (010, 100).

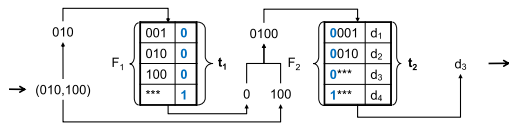


Figure 2: Pipelined lookup

### 4. PACKING APPROACH

The packing approach further reduces TCAM space consumption by allowing multiple rules from different TCAM tables to co-reside in the same TCAM entry. The packing approach is orthogonal to the multi-lookup and pipelined-lookup approaches in that it can be combined with the two approaches to further improve TCAM space efficiency. The TCAM packing idea is based on the following three key observations. First, the reconfigurability of TCAM widths is limited. TCAM chips typically only allow entry widths of 36, 72, 144, or 288 bits. This leads to wasted space as typical TCAM tables rarely can be configured to exactly one of these widths. In the standard single-lookup approach, up to 40 bits might be unused given a predicate width of 104 bits. Second, the multi-lookup and pipelined-lookup approaches produce “thin” tables of varying widths. We say the tables are thin because each table focuses on a single packet field. Thus, the table widths are much smaller than 104 bits. The widths vary because the packet fields have different lengths: 8, 16, and 32, and these predicate bits form a significant fraction of each table entry. Having multiple fields of varying widths provides opportunities to create entries that fit better in the standard TCAM widths. Third, the search key

for TCAM chips can be ternary. The global mask register (GMR) allows multiple entries from different lookup tables to co-reside in the same TCAM entry without conflict.

A natural method for packing lookup tables is to divide the TCAM into several columns and store tables in each column. We call this method *strict partitioning* since we can partition tables among these columns; however, we also show that finding an optimal partitioning is NP-Complete.

On the other hand, if we view rows as the primary dimension of TCAM chips, we can pack tables within fixed height rows. This change in perspective, which we call *shadow packing*, provides two advantages. First, tables that have different bit-widths can be better accommodated because we no longer have a vertical column boundary to maintain as shown in Figure 3(a). Second, we can reuse the table ID bits to identify tables that are packed horizontally. Figure 3(b) contains a packing tree, which determines each table's ID once the tables have been shadow packed.

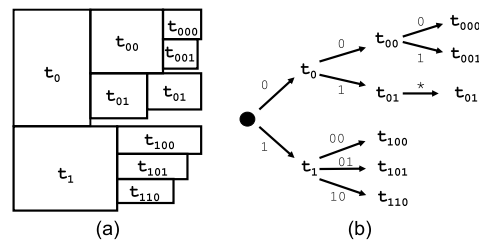


Figure 3: Shadow packed tables and packing tree

## 5. EXPERIMENTAL RESULTS

We measured the *compression ratio* of each approach for 18 real-life packet classifiers. The compression ratio of a classifier  $f$  is defined as the ratio of TCAM bits used by a particular approach  $A$  to store  $f$  in a series of TCAM chips over the TCAM bits used to store  $f$  when  $f$  is prefix expanded. Our experimental results show the average compression ratios of multi-lookup, multi-lookup with shadow packing, pipelined-lookup, and pipelined-lookup with shadow packing are 0.170, 0.048, 0.093, and 0.050, respectively. The average compression ratio for an approach  $A$  for a set of classifiers  $S$  is defined as the mean of the compression ratio of  $A$  for all the classifiers  $f$  in  $S$ . Figure 4 contains the distribution of the compression ratios for the real-life classifiers. As the figure shows, over 70% of the classifier have compression ratios below 0.0125. The key contributor to these compression ratios is the efficient utilization of TCAM bits in our approaches.

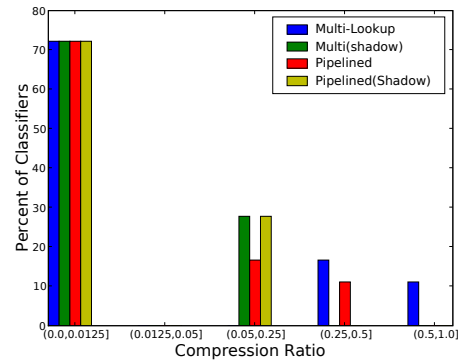


Figure 4: Distribution of compression ratios