

How Much TCAM do we Need for Splitting Traffic?

Yaniv Sadeh
Tel-Aviv University
yanivsadeh@mail.tau.ac.il

Ori Rottenstreich
Technion
or@technion.ac.il

Haim Kaplan
Tel-Aviv University
haimk@post.tau.ac.il

ABSTRACT

Traffic splitting is a required functionality in networks, for example for load balancing over paths or servers or by the source's access restrictions. The capacities of the servers (or the number of users with particular access restrictions) determine the sizes of the parts into which traffic should be split. A recent approach implements traffic splitting within the ternary content addressable memory (TCAM), which is often available in switches. It is important to reduce the amount of memory allocated for this task since TCAMs are power consuming and are often also required for other tasks such as classification and routing. Recent works suggested algorithms to compute a smallest implementation of a given partition in the longest prefix match (LPM) model. In this paper we analyze properties of such minimal representations and prove lower and upper bounds on their size. The upper bounds hold for general TCAMs, and we also prove an additional lower-bound for general TCAMs.

CCS CONCEPTS

• **Networks** → **Traffic engineering algorithms; Programmable networks; Network management.**

KEYWORDS

TCAM, traffic splitting, packet processing, load balancing.

ACM Reference Format:

Yaniv Sadeh, Ori Rottenstreich, and Haim Kaplan. 2021. How Much TCAM do we Need for Splitting Traffic?. In *The ACM SIGCOMM Symposium on SDN Research (SOSR) (SOSR '21)*, October 11–12, 2021, Virtual Event, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3482898.3483367>

1 INTRODUCTION

In many networking applications, traffic has to be split into multiple possible targets. For example, this is required in order to partition traffic among multiple paths to a destination based on link capacities, and when sending traffic to one of multiple servers proportionally to their CPU or memory resources for load balancing. Traffic splitting also arises in maintaining access-control lists (ACLs). Here, we want to limit the number of users with specific permissions. We do this by associating a fixed quota of W -bit identifiers with each ACL, and granting a particular access only to users that have one of these identifiers. In general, we address any scenario where traffic should be split by allocating a particular subset of identifiers of a

specified size to each part (a part could be associated with a server or an ACL or with some other object).

It is increasingly common to rely on network switches to perform the split [1],[20],[10]. Equal cost multipath routing (ECMP) [9] uses hashing on flows to uniformly select one of target values written as memory entries. WCMP [27],[6] (Weighted ECMP) generalizes the selection for non-uniform selections through entry repetitions, implying a distribution according to the number of appearances of each possible target. While the above works studied a fixed output partition, in a dynamic scenario mapping has to be updated following changes in traffic distribution. [3],[7],[12],[26] considered such updates for load balancing over multiple paths. They suggested update schemes that reduce transient negative impact of packet reordering, overflow (due to inaccurate partitioning of traffic), and entries re-mapping.

The implementation of some distributions in WCMP may require a large hash table. While for instance implementing a 1:2 ratio can be done with three entries (one for the first target and two for the second), the implementation of a ratio of the form $1 : 2^W - 1$ is expensive, requiring 2^W entries. Memory can grow quickly for particular distributions over many targets, even if they are only being approximated. A recent approach [10] refrains from memory blowup by comparing the hash to range-boundaries. Since the hash is tested sequentially against each range, it restricts the total number of load-balancing targets.

Recently, a natural approach was taken to implement traffic splitting within the Ternary Content Addressable Memory (TCAM), available in commodity switch architectures. For some partitions this allows a much cheaper representation [25],[14],[22],[23]. In particular, a partition of the form $1 : 2^W - 1$ can be implemented with only two entries. Unfortunately, TCAMs are power consuming and thus are of limited size [2],[18]. Therefore a common goal is to minimize the representation of a partition in TCAMs. Moreover, it is important for a network designer to know the memory cost: (i) To estimate whether a requested partition can be implemented in available TCAM quota; (ii) To prioritize partitions, or approximate-partitions, which are cheaper to represent.

The work of [25] considered only restricted TCAM encodings in which rules are *disjoint*. Since TCAMs allow overlapping rules and resolve overlaps by ordering the rules, this early approach does not take full advantage of the TCAM. Finding a representation of a partition becomes more difficult when the number of possible targets is large. Focusing on the *Longest Prefix Match model* (LPM), [14] suggested an algorithm named *Niagara*, and showed that it produces small representations in practice. They also evaluated a tradeoff of reduced accuracy for less rules. [23] suggested an optimal algorithm named *Bit Matcher* that computes a smallest TCAM for a target partition. They also proved that Niagara always computes a smallest TCAM, explaining its good empirical performance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SOSR '21, October 11–12, 2021, Virtual Event, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-9084-2/21/10...\$15.00

<https://doi.org/10.1145/3482898.3483367>

Although both Bit Matcher and Niagara compute a smallest LPM TCAM table for a given partition, no analysis of the size of this representation has been provided. An exception is the case of $k = 2$ targets for which the size of the smallest representation was analyzed in [22]. In this paper we address this gap and prove upper and lower bounds on the size of the smallest LPM TCAM needed to represent a given partition. We also give a lower bound on the minimum size of any TCAM¹ needed to represent a given partition. Note that any upper for LPM TCAMs is an upper bound for general TCAMs. The optimization problem for general TCAMs, that is, how to find a smallest or approximately smallest TCAM for a given partition, is open.

Our Contributions. (1) We prove two upper and two lower bounds on the size of the smallest TCAM for a given partition over any number k of targets. The first upper and lower bounds are general and hold for all partitions of 2^W into k parts. They are derived through new analysis of the Bit Matcher algorithm [23]. The two additional upper and lower bounds are partition-specific and consider the particular values $p_1, \dots, p_k$ of the k parts. The upper bound holds for LPM TCAMs. The lower bound has two versions, stronger for LPM TCAMs and weaker for general TCAMs.

(2) We demonstrate the tightness of our LPM bounds by constructing partitions with minimal TCAM representations that match them. We also provide examples showing that a general TCAM implementation for a partition may be strictly smaller than the best LPM implementation.

(3) The partition-specific LPM bounds are a 2-approximation. That is, the upper bound is at most twice larger than the lower bound. We evaluate how tight our bounds are experimentally.

2 TRAFFIC SPLITTING PROBLEM

A Ternary Content Addressable Memory (TCAM) of width W is a table of entries, or *rules*, each containing a *pattern* and a *target*. We assume each target is an integer in $\{1, \dots, k\}$, and also define a special target 0 for dealing with addresses that are not matched by any rule. Each pattern is of length W and consists of bits (0 or 1) and wildcards (*). An *address* is said to match a pattern if all of the specified bits of the pattern (ignoring wildcards) agree with the corresponding bits of the address. If several rules fit an address, the first rule applies. An address v is associated with the target of the rule that applies to v . We consider only minimal sets of rules, i.e. such that we cannot remove a rule without changing the mapping defined by the set. We refer to a set of TCAM rules simply as a TCAM.

Given a desired partition $P = [p_1, \dots, p_k]$ of the whole address space of 2^W addresses of W -bits such that p_i addresses should reach target i ($\sum_i p_i = 2^W$), the objective is to determine a smallest set of TCAM rules that partition traffic according to P . We assume that all the weights are positive, otherwise, any zero-weight target can be ignored, effectively reducing the number of actual targets. Therefore we have that $k \leq 2^{W/2}$.

¹Henceforth, “general TCAM” or plain “TCAM” will refer to the general case, and “LPM TCAM” will refer to the restricted case.

²The model assumes implicitly that every address is equally likely to arrive, therefore the TCAM implementation only requires each target to receive a certain number of addresses. This assumption might not hold in practice, but it can be mitigated by ignoring bits which are mostly fixed like subnet masks etc. For example, [13] analyzed

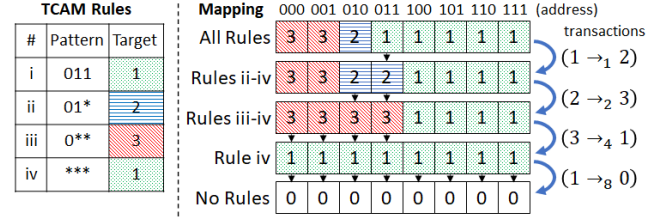


Figure 1: Example 1: transactions due to remapping.

We identify a TCAM T with a sequence s of transactions between targets defined as follows. Start with an empty sequence, and consider the change in the mapping defined by T when we delete the first rule of T , with target $i \in \{1, \dots, k\}$. Following this deletion some of the addresses may change their mapping to a different target, or become unallocated. If by deleting this rule, m addresses are re-mapped from i to $j \in \{0, \dots, k\}$ (recall that $j = 0$ means unallocated), we add to s a transaction $(i \rightarrow_m j)$. In general, one deleted rule can cause addresses to re-map to several new targets, so we may get a few transactions per deletion. Moreover, some of the addresses may be remapped from i to i so we can ignore transactions of the form $(i \rightarrow_m i)$. We then delete the second rule of T and add the corresponding transactions to s , and continue this way until T is empty and all addresses are unallocated.

DEFINITION 1 (TRANSACTIONS). We denote a transaction of size m from p_i to p_j by $(i \rightarrow_m j)$. Applying a transaction updates the values as follows: $p_i \leftarrow p_i - m$, $p_j \leftarrow p_j + m$.

EXAMPLE 1. Consider the following TCAM rules, in order of priority: $\{011 \rightarrow 1, 01* \rightarrow 2, 0** \rightarrow 3, *** \rightarrow 1\}$, note that $W = 3$. They partition $2^W = 8$ addresses to $k = 3$ targets. Deleting the first rule corresponds to the transaction $(1 \rightarrow_1 2)$. The deletion of each of the following three rules also corresponds to a single transaction, $(2 \rightarrow_2 3)$, $(3 \rightarrow_4 1)$ and $(1 \rightarrow_8 0)$, respectively, see Fig. 1.

EXAMPLE 2. To see that a deletion may correspond to multiple transactions, consider the following TCAM rules in order of priority: $\{0* \rightarrow 1, *0 \rightarrow 2, *1 \rightarrow 3\}$. Deleting the first rule corresponds to the transactions $(1 \rightarrow_1 2)$ and $(1 \rightarrow_1 3)$.

This paper mostly focuses on the Longest Prefix Match (LPM) model in which wildcards appear only as a consecutive suffix of the pattern. In general, much less is known about general TCAM rules. We do not have tighter upper bounds for general TCAMs, but we do provide a specific lower bound in Theorem 7.

The LPM model is motivated by specialized hardware as in [15], and has been studied intensively [25],[14],[22],[23]. As detailed in [11], common programmable switch architectures such as the Reconfigurable Match Tables (RMT) and Intel’s FlexPipe have tables of different types and in particular they have tables dedicated to longest prefix match rules [5],[19]. In this setting a pattern is in fact a *prefix* of bits that matches all addresses that start with this prefix. Moreover, the sets of addresses associated with each prefix form a laminar set system, and therefore the sequence of transactions that correspond to a TCAM of prefix rules is special: The deletion of every rule corresponds to a single transaction, and the size of each of these transactions is a power of 2 (revisit Example 1).

traces of real-data and concluded that for those traces about 6–8 bits out of the client’s IPv4 address are “practically uniform”.

[23] proves that both algorithms Bit Matcher and Niagara, given a partition P , compute a shortest sequence of transactions that can be mapped to a smallest TCAM table for P . We have one rule in this TCAM for each transaction in the sequence.³ Moreover, the proofs show how any shortest sequence of transactions that corresponds to P can be converted to a sequence of the same length generated by one of these algorithms. Therefore, we analyze the size of the smallest LPM TCAM by analyzing the length of a shortest sequence that “zeroes” an input partition $P = [p_1, \dots, p_k]$, where $\sum_i p_i = 2^W$. Note that transactions between non-zero indices maintain the sum, but once we zero all coordinates except one we allow a transaction with the special index 0 that corresponds to the (deletion of the) match-all TCAM rule, which zeroes the partition.

We conclude this section with an important definition, a concise summary of our results in Table 1,⁴ and two important remarks.

DEFINITION 2 (COMPLEXITY). Let $P = [p_1, \dots, p_k]$ be a partition of 2^W . We define $n(P)$ to be the size of the smallest general TCAM that realizes P . We also define $\lambda(P)$ to be the length of a shortest sequence of transactions of sizes that are powers of 2, that zeroes P . This is also equal to the size of a smallest LPM TCAM that realizes P . We say that $\lambda(P)$ is the complexity of P .

Remark 3	$\lambda(P) \geq n(P) \geq k$
Theorem 1	$k = 2 : \lambda(P) \leq \frac{1}{2}W + 2$ $k \geq 3 : \lambda(P) \leq \frac{1}{3}k(W - \lfloor \lg k \rfloor + 4)$
Theorem 2	$k = 2 : \exists P \text{ s.t. } \lambda(P) = \lceil \frac{W}{2} \rceil + 1$
Theorem 3	$k = 3 : \exists P \text{ s.t. } \lambda(P) = W + 1$
Theorem 4	$k \geq 4 : \exists P \text{ s.t. } \lambda(P) \leq \lfloor \frac{k-1}{3} \rfloor (W - \lfloor \lg k \rfloor + 1)$
Theorem 5	$\lceil \frac{S(P)+1}{2} \rceil \leq \lambda(P) \leq S(P) + 1 - M(P)$
Theorem 6	$\exists P \text{ s.t. Theorem 5 is tight (UB/LB/both)}$
Theorem 7	Assume $ \phi(p_1) \geq \phi(p_2) \geq \dots \geq \phi(p_k) $, then $n(P) \geq \max_{i=1, \dots, k} \lg(\phi(p_i) + 1) + i - 1$
Corollary 8	$\exists P \text{ s.t. } n(P) \geq \lg(W - \lfloor \lg k \rfloor + 3) + k - 2$

Table 1: Summary of results regarding $n(P)$ and $\lambda(P)$ in terms of k and W , or $\phi(p_i)$, $S(P)$ and $M(P)$ as in Definitions 5-6.

REMARK 1 (GENERAL TCAMS). No non-trivial algorithm to compute exactly or to approximate a smallest general TCAM for a given partition is known.⁵ On the other hand, there are also no hardness proofs for this problem. The LPM model is much simpler because every rule corresponds to exactly one transaction. Indeed, using general rules sometimes allows a smaller TCAM. For the following two partitions $\lambda(P) > n(P)$.

The partition $P = [4, 3, 3, 3, 3]$ satisfies $\lambda(P) = 7$. But $n(P) = 5$ as follows: $\{**00 \rightarrow 1, 00** \rightarrow 2, 01** \rightarrow 3, 10** \rightarrow 4, 11** \rightarrow 5\}$. Deleting the first rule corresponds to 4 transactions (one-to-many): $(1 \rightarrow_1 i)$ for $i = 2, 3, 4, 5$.

³The translation works by processing the transactions ordered from largest to smallest size. We allocate a rule for i with a prefix that extends a prefix of a rule of j 's when we process a transaction $(i \rightarrow_{2^h} j)$. For more details see [23].

⁴Throughout the paper, $\lg$ is the logarithm in base 2.

⁵If the mapping is a function, i.e. each address has a predefined target, then there are exponential algorithms, see [17].

Using general rules can reduce the size of a TCAM even for $k = 2$. For $W = 10$, to represent $P = [683, 341]$ we need $\lambda(P) = 6$ LPM rules (see Theorem 2). But $n(P) \leq 5$: $\{0000000000 \rightarrow 2, *000**000 \rightarrow 1, **000***** \rightarrow 2, 00***** \rightarrow 2, ***** \rightarrow 1\}$.

REMARK 2 (“PRACTICAL TAKE-AWAY”). The size bounds which we prove should help a network engineer to better understand the required number of TCAM rules, as a function of the number of targets which is architecture dependent, and the choice of the address-space by which to split the traffic (its bits-size).

Table 1 summarizes the needed knowledge. The most relevant are the upper bounds of Theorem 1 and Theorem 5. In practice, by simulations, the complexity of “an average partition” is better from the upper bound of Theorem 1 only by a small constant factor. This means that we can reduce the the number of rules, by reducing the number of targets, k , and the width of the space W . As an example of saving space at the cost of accuracy, revisit $P = [683, 341]$ in Remark 1. We can represent instead $P' = [672, 352]$ using only the last 3 rules out of the 5 which we used above to represent P . Note that its effective width is in fact $W' = 5$ because $P' = 32 \cdot [21, 11]$, so when ignoring the scaling factor, the rules actually induce the partition $[21, 11]$.

3 TCAM SIZE BOUNDS

In this section we prove upper and lower bounds on the complexity $\lambda(P)$ of a partition P as a function of the number of targets k and the sum 2^W (TCAM width W).

REMARK 3 (TRIVIAL LOWER BOUND). For any partition P to k targets we have that $n(P) \geq k$. This is because each target must be associated with at least one rule. Also, $\lambda(P) \geq n(P)$ because LPM rules are more restrictive.

The remaining analysis in this section focuses on the upper bound, and is based on the properties of sequences which compute an optimal (minimal size) LPM TCAM for a given partition as generated by the Bit Matcher algorithm [23] described in Algorithm 1. Observe that the transactions are generated in an increasing order of size. The analysis relies on the bit-lexicographic order used by Bit Matcher.

DEFINITION 3 (BIT-LEVEL). Let n be a non-negative integer. We denote by $n[\ell]$ the ℓ -th bit in the binary representation of n . $n[0]$ is the least-significant bit of n . We say that $n[\ell]$ is the bit of n at **level** ℓ .

DEFINITION 4 (BIT-LEXICOGRAPHIC ORDER). Let x, y be non-negative integers. We say that $x <_{\text{lex}} y$ or that x is **bit-lexicographic** smaller than y if $x^r < y^r$ where x^r and y^r are obtained from x and y by reversing their binary representations with respect to a fixed word size, respectively.

For example, $5 <_{\text{lex}} 3$ (as $5_{10} = 101_2$, $3_{10} = 011_2$). Also, every even number is smaller than any odd number.

The first lemma refers to changes in the binary representation of the weights following an application of a transaction.

LEMMA 1. Let $P = [p_1, \dots, p_k]$ be a partition of 2^W , and let s be a sequence of transactions generated for P by Bit Matcher (see Algorithm 1). The following two properties hold:

(i) Consider a specific transaction $(i \rightarrow_{2^\ell} j)$ after all smaller transactions have been applied. Let z denote the number of consecutive

Algorithm 1: Bit Matcher [23] (rephrased)

Input: A partition $P = [p_1, \dots, p_k]$, each p_i is a non-negative integer and $\sum_{i=1}^k p_i = 2^W$.

Output: Shortest sequence s of transactions of sizes which are powers of 2, that zeroes P .

Initialize s to be an empty sequence.

for level $d = 0 \dots W - 1$ **do**

Let A be the set of weights $i \geq 1$ where $p_i[d] = 1$.

Let A_h consists of the $|A|/2$ largest weights in A in bit lexicographic order, and let $A_l = A \setminus A_h$.

Pair the elements of A_h and A_l arbitrarily and for each pair $p_i \in A_l, p_j \in A_h$ append to s and apply to P the transaction $(i \rightarrow_{2^d} j)$.

end

Let i be such that $p_i = 2^W$. Add to s the transaction $(i \rightarrow_{2^W} 0)$, **return** s .

bits in levels $\geq \ell$ of p_i and p_j that are guaranteed to be zero (and stay zero) following this transaction.⁶ If $\ell < W - 1$, then $z \geq 3$. Moreover, if $k = 2$ then $z \geq 4$.

(ii) Zeroed bits associated with different transactions are different, and the bits that are zeroed by $(i \rightarrow_{2^\ell} j)$ are consecutive from $p_i[\ell]$ and $p_j[\ell]$ upwards, until the next level in which i and j participate in a transaction (this level may be different for i and j).

EXAMPLE 3. Assume that $p_i = 5$ and $p_j = 3$ and that we apply the transaction $(i \rightarrow_1 j)$. Then the weights change to $p_i = p_j = 4$, whose binary representation is 100. The zeroed bits are underlined, and in this example, $z = 4$ (two in i and two in j).

PROOF OF LEMMA 1. Bit Matcher applies a transaction at level ℓ between targets i and j with bit ℓ equals 1. It follows that the first two bits that are guaranteed to be zero following this transaction are $p_i[\ell]$ and $p_j[\ell]$.

Since Bit Matcher applies the transaction in bit-lexicographic order, we have $p_i[\ell+1] \leq p_j[\ell+1]$ (before the transaction is applied). Consider the three possible cases for $p_i[\ell+1]$ and $p_j[\ell+1]$ before the transaction is applied.

- (1) $p_i[\ell+1] = 0$ and $p_j[\ell+1] = 0$: following the transaction, we have $p_i[\ell+1] = 0, p_j[\ell+1] = 1$.
- (2) $p_i[\ell+1] = 0$ and $p_j[\ell+1] = 1$: following the transaction, we have $p_i[\ell+1] = 0, p_j[\ell+1] = 0$ (due to carry).
- (3) $p_i[\ell+1] = 1$ and $p_j[\ell+1] = 1$: following the transaction, we have $p_i[\ell+1] = 1, p_j[\ell+1] = 0$ (due to carry).

Overall, we see that $z \geq 3$ ($p_i[\ell] = p_j[\ell] = 0$ and at least another bit in level $\ell+1$). When $k = 2$, only Case (2) is possible, because $p_1 + p_2 = 2^W$, so we get $z \geq 4$. Property (ii) of the claim is trivial by the way Bit Matcher works. $\square$

Now we use Lemma 1 to prove a worst-case upper bound. That is, the largest possible minimum-size LPM TCAM for a partition of 2^W addresses to k targets.

THEOREM 1 (UPPER BOUND). Let P be a partition of 2^W to k parts.

- If $k = 2$: $\lambda(P) \leq \frac{1}{4}k(W - \lfloor \lg k \rfloor + 1) + k = \frac{1}{2}W + 2$.
- If $k \geq 3$: $\lambda(P) \leq \frac{1}{3}k(W - \lfloor \lg k \rfloor + 1) + k = \frac{1}{3}k(W - \lfloor \lg k \rfloor + 4)$.

⁶Note that levels $0, \dots, \ell - 1$ are known to be zero prior to the transaction.

PROOF. We argue first for $k \geq 3$ and then consider the case $k = 2$. Consider the binary representation of the weights $p_1, \dots, p_k$. By Lemma 1, each transaction of Bit Matcher at level ℓ can be associated with (at least) 3 bits at levels ℓ and $\ell + 1$ that remain zero after it is applied. Furthermore, bits associated with different transactions are different.

Let $N = W - \lfloor \lg k \rfloor$. Since there are at most $k(N + 1)$ bits in the $N + 1$ least significant levels (in total for all targets), and we associated uniquely three bits with each transaction, then we can have at most $\frac{1}{3}k(N + 1)$ transactions at the N least significant levels. Following these transactions the N least significant levels of $p_1, \dots, p_k$ are all zero. We considered $N + 1$ levels since transactions at the first N levels may “charge” bits at the $N + 1$ level.

After these transactions have been applied, we observe that no more than k bits are 1 in the (current) binary representations of all p_i , for $i \geq 1$. Indeed, each 1 bit at the top $\lfloor \lg k \rfloor$ levels contributes at least $2^{W - \lfloor \lg k \rfloor} \geq \frac{2^W}{k}$ to the sum of all p_i , which equals 2^W . Thus, Bit Matcher performs at most k additional transactions at the most significant $\lfloor \lg k \rfloor$ levels. In total we get that for any partition P with $k \geq 3$: $\lambda(P) \leq \frac{1}{3}k(W - \lfloor \lg k \rfloor + 1) + k$.

When $k = 2$, we repeat the same argument. The only difference is that each transaction at level $\leq N$ is associated with (at least) 4 zeroed bits (by Lemma 1), so the factor of $\frac{1}{3}$ in the bound changes to $\frac{1}{4}$. $\square$

Next, we show worst-case partitions matching the upper bounds on $\lambda(P)$ in Theorem 1 up to minor additive constants.

THEOREM 2 ($k = 2$). Let $x = \text{round}\left(\frac{2^W}{3}\right)$ for $W \geq 1$. The partition $P = [x, 2^W - x]$ satisfies $\lambda(P) = \lceil \frac{W}{2} \rceil + 1$.

PROOF. The binary expansion of $\frac{1}{3} = 0.0101\dots$ is infinite with alternation of 0s and 1s. The binary representations of $\frac{2^W}{3}$ and $\frac{2 \cdot 2^W}{3}$ are both shifts of this representation, so modulo 2 one of them is 0.1010... and the other is 1.0101.... In both cases the rounding makes both x and $2^W - x$ odd, so both x and $2^W - x$ have alternating and complementing bit representations except for the least significant bit in which both have 1. The Bit Matcher algorithm applies $\lceil \frac{W}{2} \rceil$ transactions, at every even level ($\ell = 0, 2, \dots$). It follows that the total number of transactions is $\lceil \frac{W}{2} \rceil + 1$, the +1 is due to the last transaction $(i \rightarrow_{2^W} 0)$ for $i = 1$ or $i = 2$. $\square$

THEOREM 3 ($k = 3$). Let $x = \lfloor \frac{2^W}{3} \rfloor$. If x is even, let $P = [x, x + 1, x + 1]$, and if x is odd let $P = [x, x, x + 1]$ (one can verify that $\sum_{i=1}^3 p_i = 2^W$ in both cases). Then $\lambda(P) = W + 1$.

Note that if we substitute $k = 3$ in Theorem 1 we get an upper bound of $W + 3$.

PROOF SKETCH. Prove by induction that there is exactly one transaction per level, the base is for $W = 1$: $\lambda([0, 1, 1]) = 2$. $\square$

THEOREM 4 ($k > 3$). There exists a partition P such that $\lambda(P) > \lfloor \frac{k-1}{3} \rfloor (W - \lfloor \lg k \rfloor + 1)$.

PROOF SKETCH. Divide the parts to $m = \lfloor \frac{k-1}{3} \rfloor$ disjoint triplets and allocate a total weight of $2^{W-1-\lfloor \lg m \rfloor}$ to each triplet as in Theorem 3. Allocate the remaining weight to the remaining parts.

The core of the proof is to show that each triplet contributes $W - \lceil \lg m \rceil > W - \lceil \lg k \rceil + 1$ transactions. Due to space constraints, we omit these details. $\square$

4 TCAM SIZE SIGNED-BITS BOUNDS

In this section we prove tighter lower and upper bounds on $\lambda(P)$. The upper bound applies to $n(P)$ as well, and we also improve the trivial lower bound of k on $n(P)$. These bounds depend also on $p_1, \dots, p_k$, rather than just on k and W . The following definition will be used throughout this section.

DEFINITION 5 (SIGNED-BITS REPRESENTATION). Every integer n can be represented by a sequence of m coefficients $a_m \dots a_0$, for an integer m , where $\forall i : a_i \in \{-1, 0, 1\}$ and $a_m \neq 0$, such that $n = \sum_{i=0}^m (a_i \cdot 2^i)$. We will denote the signed-digit -1 by $\bar{1}$.

For example the standard binary representation of a non-negative integer n is also a signed-bits representation of n . Such a representation is in general not unique but there is a unique representation in which no two consecutive coefficients a_i and a_{i+1} are non-zero. We denote this representation by $\phi(n)$. We use the notations $\phi(n)[\ell]$ to refer to the value of a_ℓ , and $|\phi(n)|$ is the number of **non-zero** coefficients.⁷

One can verify that $\phi(n) = n$ for $|n| \leq 1$, and: $\phi(2n) = \phi(n) \circ 0$, $\phi(4n+1) = \phi(2n) \circ 1$, $\phi(4n-1) = \phi(2n) \circ \bar{1}$ where $\circ$ stands for concatenation.

The paper [4] uses the generalized definition of signed-digits representation to speed-up arithmetic operations. An alternative view of signed-digits representation is representing an integer as the difference of two non-negative integers. [21] uses this alternative framing to investigate binary arithmetic. Overall, signed-bits are of interest since they come up in minimization/optimization scenarios, see [16, Section 6] for a survey. In the context of our work, [22] used this representation to give an exact expression for $\lambda(P)$ when $k = 2$. We study the case of an arbitrary number $k \geq 2$ of targets and in particular that of $k \geq 3$.

LEMMA 2. For integers n and h : $|\phi(n + 2^h)| - |\phi(n)| \leq 1$.

PROOF. For a positive integer d , let $||d||$ denote the number of 1 bits in its binary representation. For every integer n define $n = n_+ - n_-$ such that $n_+ = \sum_{a_i=1} 2^i$ and $n_- = \sum_{a_i=\bar{1}} 2^i$ where a_i are the coefficients in $\phi(n)$. We have $|\phi(n)| = ||n_+|| + ||n_-||$. We rely on the following fact, proven in [8, Section 4]: If $n = x - y$ for non-negative integers x, y , then $|\phi(n)| \leq ||x|| + ||y||$.

Now, denote $m = n + 2^h$. Then $n = m - 2^h = m_+ - (m_- + 2^h)$, and therefore: $|\phi(n)| \leq ||m_+|| + ||m_- + 2^h|| \leq ||m_+|| + ||m_-|| + ||2^h|| = |\phi(m)| + 1$, where the second inequality is by counting "standard" bits (it may be a strict inequality due to potential carry). The other direction is similar: $|\phi(m)| \leq ||n_+ + 2^h|| + ||n_-|| \leq |\phi(n)| + 1$. Together: $|\phi(n + 2^h)| - |\phi(n)| \leq 1$. $\square$

DEFINITION 6. Let $P = [p_1, \dots, p_k]$ be a vector of integers. We define the sum and max signed-bits of its values as: $S(P) = \sum_{i=1}^k |\phi(p_i)|$ and $M(P) = \max_{i \in [k]} |\phi(p_i)|$.

THEOREM 5. For any partition P , $\left\lceil \frac{S(P)+1}{2} \right\rceil \leq \lambda(P) \leq S(P) + 1 - M(P)$.

⁷The sequence $|\phi(n)|$ for $n \geq 0$ is known as <https://oeis.org/A007302>.

PROOF. Define the vector X such that $x_0 = -2^W$ and for $i \in [k]$: $x_i = p_i$. Then every sequence s that zeroes P also zeroes X because the excess of 2^W weight from indices 1 to k is transacted to index 0. Each transaction $(i \rightarrow_{2^\ell} j)$ has a size that is a power of two, so by Lemma 2 it can decrease the total number of non-zero signed-bits in the representation of X by at most 2, one in the representation of x_i and one in the representation of x_j . We need to zero $S(X) = S(P) + |\phi(-2^W)| = S(P) + 1$ signed-bits, so any sequence that zeroes X must have at least $\frac{S(P)+1}{2}$ transactions. This proves that $\left\lceil \frac{S(P)+1}{2} \right\rceil \leq \lambda(P)$.

let $j = \operatorname{argmax}_{i \in [k]} |\phi(x_i)|$ be the "anchor index". We can zero its bits "for free" by taking care of all other indices as follows: for $i \neq j$, if $\phi(x_i)[\ell] = 1$ we apply the transaction $(i \rightarrow_{2^\ell} j)$, and if $\phi(x_i)[\ell] = \bar{1}$ we apply the transaction $(j \rightarrow_{2^\ell} i)$. These transactions zero every x_i for $i \neq j$, and since $\sum_{i=0}^k x_i = 0$ is an invariant, we get that the sequence also zeroes x_j . Overall, this sequence requires $S(P) + 1 - M(P)$ transactions, and this proves $\lambda(P) \leq S(P) + 1 - M(P)$. $\square$

REMARK 4. The upper bound in Theorem 5 can be improved further, by changing the anchor index throughout the levels. If $\phi(x_{j_0})$ is dense in non-zero signed-bits in the lower levels, say up to level d_0 , then we can pick j_0 as the anchor and generate the transactions of sizes at most 2^{d_0} against it. Then, we can decide to change the anchor to j_1 for all the transactions up to size 2^{d_1} (for $d_1 > d_0$), and so on. In every anchor-change, we might have to add another transaction due to carry that accumulates in the current anchor. Because of this carry, it is not necessarily beneficial to switch anchors too frequently. However, it is likely that by dividing W into more than one bulk of levels we can improve the upper bound. That being said, the expression $S(P) + 1 - M(P)$ has a simple closed-form and is already a 2-approximation.

THEOREM 6. The bounds of Theorem 5 are tight. More formally, there exist partitions such that:

- (1) $\left\lceil \frac{S(P)+1}{2} \right\rceil < \lambda(P) = S(P) + 1 - M(P)$ (tight UB).
- (2) $\left\lceil \frac{S(P)+1}{2} \right\rceil = \lambda(P) < S(P) + 1 - M(P)$ (tight LB).
- (3) $\left\lceil \frac{S(P)+1}{2} \right\rceil = \lambda(P) = S(P) + 1 - M(P)$ (sandwiched). In particular, for $k = 2$ this is always the case.

PROOF. The upper bound is tight for any partition in which each transaction except the last can zero at most one signed-bit. For example, any partition where the signed-bits representation of every weight does not have any coefficient that is $\bar{1}$, e.g. $P = [5, 5, 5, 1]$ (with $W = 4$) has $\lambda(P) = 6$ by $(4 \rightarrow_1 3)(2 \rightarrow_1 1)(3 \rightarrow_2 1)(3 \rightarrow_4 2)(2 \rightarrow_8 1)(1 \rightarrow_{16} 0)$. Since $\phi(P) = [101, 101, 101, 1]$, we have $S(P) = 7$, $M(P) = 2$. Here the lower bound is not tight.

The lower bound is tight for any partition in which each transaction zeroes two signed-bits, e.g. $P = [1, 3, 12]$ for which $\lambda(P) = 3$ by $(1 \rightarrow_1 2)(2 \rightarrow_4 3)(3 \rightarrow_{16} 0)$. Since $\phi(P) = [1, 10\bar{1}, 10\bar{1}00]$, we have $S(P) = 5$, $M(P) = 2$. Here the upper bound is not tight.

The partitions in which the bounds are equal are those where a single weight participates in all the transactions, the anchor, and each transaction zeroes two signed-bits. For example, $P = [15, 4, 45]$ (with $W = 6$): $\lambda(P) = 4$ by $(3 \rightarrow_1 1)(2 \rightarrow_4 3)(1 \rightarrow_{16} 3)(3 \rightarrow_{64} 0)$. Since $\phi(P) = [1000\bar{1}, 100, 10\bar{1}0\bar{1}01]$, we have $S(P) = 7$, $M(P) = 4$. In particular, when $k = 2$ this is always the case (originally shown by

[22]): Denote $m_i = |\phi(p_i)|$ and $m = \min(m_1, m_2)$, then the upper bound is $S(P) + 1 - M(P) = m + 1$. Because $p_2 = 2^W - p_1$ and $|\phi(p_2)| = |\phi(-p_2)|$, by Lemma 2, $|m_1 - m_2| \leq 1$, so no matter whether $m_1 = m_2 = m$ or $|m_1 - m_2| = 1$, we get that the lower bound also equals $\lceil \frac{S(P)+1}{2} \rceil = m + 1$. $\square$

The following theorem gives a lower bound on the minimal size of a general TCAM. We omit its proof due to lack of space.

THEOREM 7. *Let P be a partition, and without loss of generality assume that $|\phi(p_1)| \geq |\phi(p_2)| \geq \dots \geq |\phi(p_k)|$. Then $n(P) \geq \max_{i=1, \dots, k} \lg(|\phi(p_i)| + 1) + i - 1$.*

We note that this lower bound is likely very loose for many partitions. However, this lower bound enables us to determine hard-partitions even for general TCAMs, despite the fact that no feasible algorithm to compute or approximate such TCAMs exist.

EXAMPLE 4. *Consider the partition $P = [683, 341]$ from Remark 1. In signed-bits, we have $p_1 = 10\bar{1}0\bar{1}0\bar{1}0\bar{1}$ and $p_2 = 101010101$. Then $|\phi(p_1)| = 6$ and $|\phi(p_2)| = 5$, and by Theorem 7 we get $n(P) \geq \max(\lg(7), \lg(6) + 1) \Rightarrow n(P) \geq 4$. This means that every TCAM that represents P must have at least 4 rules.*

COROLLARY 8. *There exists a partition P of 2^W to k parts that satisfies $n(P) \geq \lg(W - \lceil \lg k \rceil + 3) + k - 2$.*

PROOF. Let $h = W - \lceil \lg k \rceil$, and define an initial partition Q of 2^W such that for every i , q_i is either 2^h or 2^{h+1} , and $\forall i : q_k \geq q_i$. Next define $\Delta = \sum_{j=1}^{\lfloor h/2 \rfloor} 2^{h-2j}$ and perturb Q to get P as follows. If k is even then $p_i = q_i + (-1)^i \Delta$. If k is odd, then $q_k = 2^{h+1}$, we define $p_i = q_i + (-1)^i \Delta$ for $i \leq k-2$, $p_{k-1} = q_{k-1} - \Delta$ and $p_k = q_k + 2\Delta$. One can verify that $\sum_{i=1}^k p_i = 2^W$. Observe that $|\phi(2^{h+1} \pm 2\Delta)| = |\phi(2^h \pm \Delta)| = |\phi(2^{h+1} \pm \Delta)| = \lfloor \frac{h}{2} \rfloor + 1$. Thus $|\phi(p_i)| = \lfloor \frac{h}{2} \rfloor + 1$ for all i , and by Theorem 7: $n(P) \geq \lg(\lfloor \frac{h}{2} \rfloor + 2) + k - 1 \geq \lg(W - \lceil \lg k \rceil + 3) + k - 2$. $\square$

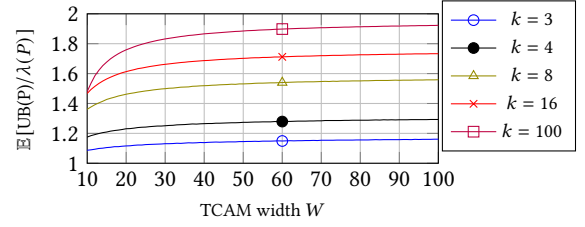
EXAMPLE 5. *Let $W = 7$ and $k = 5$. Then $h = 4$, $\Delta = 2^2 + 2^0 = 5$, and $Q = [16, 16, 32, 32, 32]$. Since k is odd, we get $P = [11, 21, 27, 27, 42] = [10\bar{1}0\bar{1}, 10101, 100\bar{1}0\bar{1}, 100\bar{1}0\bar{1}, 101010]$. $\forall i : \phi(p_i) = 3$, and $n(P) \geq 6$.*

5 EXPERIMENTAL RESULTS

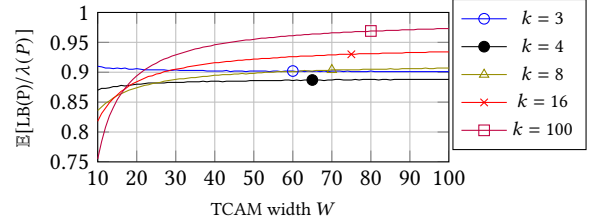
In this section we evaluate experimentally the upper and lower bounds of Theorem 5. While these signed-bits bounds are partition-specific, the natural parameters with which we are working are k (number of parts) and W (sum of 2^W). For all partitions with the same k and W , we estimated the average ratio of our signed-bits bounds and the true size of the smallest LPM TCAM.

We sample many ordered-partitions for fixed k and W , and average over all of them. An ordered partition is a partition where the order of the parts matters, e.g. $[1, 3] \neq [3, 1]$. We sample uniformly ordered-partitions P as follows: Choose uniformly a subset of $k-1$ values $B \subset \{1, \dots, 2^W - 1\}$, denote the i^{th} smallest value in B by b_i and define $b_0 = 0$ and $b_k = 2^W$. The i^{th} part of the partition is $b_i - b_{i-1} > 0$.

For each partition, we estimate how good the bounds are by computing the ratios $\frac{\lceil (S(P)+1)/2 \rceil}{\lambda(P)}$ and $\frac{S(P)+1-M(P)}{\lambda(P)}$. Then, we compute the (numeric) expectation of these values by averaging over 10,000 sampled partitions per pair of the parameters (k, W) .



(a) Upper Bound, $UB(P) \equiv S(P) + 1 - M(P)$



(b) Lower Bound, $LB(P) \equiv \lceil (S(P) + 1)/2 \rceil$

Figure 2: Estimating the bounds of Theorem 5.

Fig. 2 plots the results for $W \in [10, 100]$ and $k = 3, 4, 8, 16, 100$. Overall, there are 10 graphs in the figure, upper and lower bounds for each of the values of k . The upper bounds are separated clearly, while the lower bounds are relatively closer together, and approximately satisfy $E[\frac{\lceil (S(P)+1)/2 \rceil}{\lambda(P)}] \approx 0.9$ when $W \geq 20$.

From this experiment, it is clear that the lower bound can be used as a good estimator to the actual value $\lambda(P)$ if W is not too small. Note that it is much easier to compute this lower bound rather than to compute $\lambda(P)$ itself since it only requires counting bits in the signed-bits representation of P 's parts.⁸ The upper bound, on the other hand, gets looser when k grows larger, which is expected by Remark 4, yet the ratio between these bounds will never exceed 2.

6 CONCLUSIONS

In this paper we thoroughly studied the size of a minimal TCAM table that implements a specific partition P of the address-space. We proved that a partition requires no more than $\frac{kW}{3}$ rules, and also analyzed bounds that depend on the signed-bits representation of a partition, and found in our simulations that our lower bound in terms of this representation provides a good estimation to $\lambda(P)$, in expectation.

The signed-bits representation of a number has less non-zero coefficients if we round it to a multiple of a large power of 2 (ignoring least significant (signed-) bits). This implies that to reduce the number of TCAM rules, at the cost of losing accuracy we can just round or ignore the least significant bits in the signed representations of the parts. Optimal solutions for reducing size at the cost of accuracy already exist [24], but our result provides an easy rule-of-thumb to estimate this tradeoff: cutting-down on the binary representation of the weights from W bits to $W' < W$ is expected to reduce the TCAM size by a factor of approximately $\frac{W'}{W}$.

⁸Counting is simpler to implement than Bit Matcher or Niagara, though in practice an implementation would be available in order to compute the TCAM rules. Counting bits is also technically quicker (negligible in practice).

REFERENCES

- [1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM*.
- [2] Michiel Appelman and Maikel de Boer. 2012. Performance analysis of OpenFlow hardware. *University of Amsterdam, Tech. Rep* (2012).
- [3] N. Sertac Artan, Haowei Yuan, and H. Jonathan Chao. 2008. A Dynamic Load-Balanced Hashing Scheme for Networking Applications. In *IEEE GLOBECOM*.
- [4] A. Avizienis. 1961. Signed-Digit Numbe Representations for Fast Parallel Arithmetic. *IRE Transactions on Electronic Computers* EC-10, 3 (1961), 389–400.
- [5] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando A. Mujica, and Mark Horowitz. 2013. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN. In *ACM SIGCOMM*.
- [6] Zhiruo Cao, Zheng Wang, and Ellen W. Zegura. 2000. Performance of Hashing-Based Schemes for Internet Load Balancing. In *IEEE INFOCOM*.
- [7] Tat Wing Chim, Kwan L. Yeung, and King-Shan Lui. 2005. Traffic distribution over equal-cost-multi-paths. *Computer Networks* 49, 4 (2005), 465–475.
- [8] Prasanna Ganesan and Gurmeet Singh Manku. 2004. Optimal Routing in Chord. In *ACM-SIAM SODA*. 176–185.
- [9] C. Hoppps. 2000. Analysis of an equal-cost multi-path algorithm. (Nov. 2000). RFC 2992.
- [10] Kuo-Feng Hsu, Praveen Tammana, Ryan Beckett, Ang Chen, Jennifer Rexford, and David Walker. 2020. Adaptive Weighted Traffic Splitting in Programmable Data Planes. In *ACM Symposium on SDN Research (SOSR)*.
- [11] Lavanya Jose, Lisa Yan, George Varghese, and Nick McKeown. 2015. Compiling Packet Programs to Reconfigurable Switches. In *USENIX NSDI*.
- [12] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur W. Berger. 2007. Dynamic load balancing without packet reordering. *Computer Communication Review* 37, 2 (2007), 51–62.
- [13] Nanxi Kang, Monia Ghobadi, J. Reumann, A. Shraer, and J. Rexford. 2014. *Niagara: Scalable Load Balancing on Commodity Switches*. Technical Report TR-973-14. Princeton.
- [14] Nanxi Kang, Monia Ghobadi, John Reumann, Alexander Shraer, and Jennifer Rexford. 2015. Efficient traffic splitting on commodity switches. In *ACM CoNEXT*.
- [15] Soraya Kasnavi, Vincent C. Gaudet, Paul Berube, and Jose N. Amaral. 2005. A hardware-based longest prefix matching scheme for TCAMs. In *IEEE International Symposium on Circuits and Systems*.
- [16] Gurmeet Singh Manku and Joe Sawada. 2005. A Loopless Gray Code for Minimal Signed-Binary Representations. In *Annual European Conference on Algorithms*.
- [17] Rick McGeer and Praveen Yalagandula. 2009. Minimizing Rulesets for TCAM Implementation. In *IEEE INFOCOM*.
- [18] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru M. Parulkar, Larry L. Peterson, Jennifer Rexford, Scott Shenker, and Jonathan S. Turner. 2008. OpenFlow: Enabling innovation in campus networks. *Computer Communication Review* 38, 2 (2008), 69–74.
- [19] Recep Ozdag. 2012. Intel®Ethernet Switch FM6000 Series-Software Defined Networking. *Intel Corporation* (2012).
- [20] Parveen Patel, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert G. Greenberg, David A. Maltz, Randy Kern, Hemant Kumar, Marios Zikos, Hongyu Wu, Changhoon Kim, and Naveen Karri. 2013. Ananta: Cloud scale load balancing. In *ACM SIGCOMM*.
- [21] George W. Reitwiesner. 1960. Binary Arithmetic. *Advances in Computers*, Vol. 1. Elsevier, 231–308.
- [22] Ori Rottenstreich, Yossi Kanizo, Haim Kaplan, and Jennifer Rexford. 2018. Accurate Traffic Splitting on Commodity Switches. In *ACM SPAA*.
- [23] Yaniv Sadeh, Ori Rottenstreich, Arye Barkan, Yossi Kanizo, and Haim Kaplan. 2020. Optimal Representations of a Traffic Distribution in Switch Memories. *IEEE/ACM Trans. Netw.* 28, 2 (2020), 930–943.
- [24] Yaniv Sadeh, Ori Rottenstreich, and Haim Kaplan. 2020. Optimal Approximations for Traffic Distribution in Bounded Switch Memories. In *ACM CoNEXT*.
- [25] Richard Wang, Dana Butnariu, and Jennifer Rexford. 2011. OpenFlow-based server load balancing gone wild. In *USENIX Hot-ICE*.
- [26] Ning Wu, Shih-Hao Tseng, and Ao Tang. 2018. Accurate Rate-Aware Flow-level Traffic Splitting. In *Allerton Conference on Communication, Control, and Computing*.
- [27] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: Weighted cost multipathing for improved fairness in data centers. In *EuroSys*.