Green Control of Network Nodes with TCAM-ALG

Mariusz Żal, Filip Idzikowski
Faculty of Electronics and Telecommunications, Poznan University of Technology, Poland, firstname.lastname@put.poznan.pl

***Abstract*—Typically Ternary Content Addressable Memories (TCAMs) are used for simple lookups inside forwarding or routing information bases in routers or switches. We propose a new application of TCAM as the element of switching fabric control module inside a switching node, which is able to directly determine the set of established connections that block an arriving connection. We propose an algorithm called TCAM-ALG that utilizes this feature when establishing and releasing connections. Usage of TCAM-ALG and no Central Processing Unit (CPU) in the proposed control module results in lower energy consumption than in case of control modules using three alternative algorithms utilizing Random-Access Memory (RAM) and a CPU.**

I. Introduction

Green networking has become a well-established field of research. There are many works that focus on reduction of energy consumption of various network components in different types of networks [1]. Most of the works in the field of wired networks focus on the data plane [2, 3]. We take a different approach and look at the control plane of the network. In particular, we look at the locally implemented control algorithm of a single network node.

Each switching node is an important element of a data transmission system such as a telecommunications network. Its task includes reliable and optimal (from the service quality point of view) data distribution to particular network areas as well as data exchange between autonomous networks [4]. A general switching node architecture is presented in Fig. 1. It is divided into a data plane and a control plane. In the data plane, each node contains a certain number of input and output interfaces through which end users in access networks or other switching nodes in backbone networks are attached. When an input interface receives a connection request, information about this event is sent to the control plane that is responsible for serving the received data. We focus on energy consumed by the control module that uses a control algorithm to make decisions how to forward data from its input interface to its output interface. The forwarding operation is done by the switching fabric, which is the central element of any switching node.

Routers and switches require high-performance searching engines to make forwarding decisions. A commercial TCAM device can perform up to 360 million searches per second [5]. Such high performance leads to increased energy consumption in comparison to RAM. Cost of TCAM (expressed for example in the number of used transistors) is also non-negligible. Therefore, a number of energy-efficient TCAM architectures have been proposed for packet-switched networks. They focus mainly on reduction of power consumption of match line [6] or energy-aware usage [5] In this work, we consider a new area of TCAM usage in the control plane, and propose a control algorithm utilizing it. We compare the proposed control algorithm with three alternative algorithms known from the literature answering the following research questions: (i) how much energy is consumed by a network node for its control?, (ii) what is the cost of memory used in a network node taking into account the difference of using TCAM and RAM?

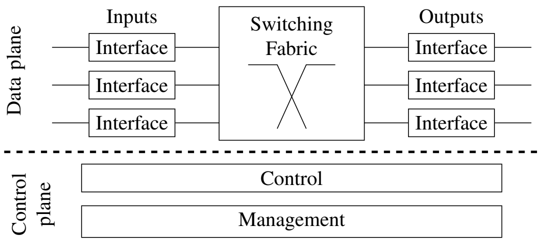


Fig. 1. General switching node architecture.

The rest of the paper is structured as follows. The node architecture is detailed in Section II. The model of the switching fabric based on De Bruijn graph properties is presented in Section III. The proposed control algorithm TCAM-ALG is described in Section IV and evaluated in Section V. Section VI concludes this work.

II. Node architecture using TCAM

We describe the data plane and the control plane of the switching node architecture introduced in Fig. 1. Table I contains a summary of notation used in this work.

A. Data plane

The switching fabric is the workhorse of the data plane. It is used to transport data (e.g., packets) from input to output interfaces. In this work, a $\log_2(N,0,p)$ switching fabric (called also a banyan-type switching fabric) with perfect-shuffle topology is considered (Fig. 2) [7]. This switching fabric is constructed with p vertically stacked $\log_2(N,0,1)$ switching fabrics, where N, being a power of 2, denotes the number of both inputs and outputs. Each copy of $\log_2(N,0,1)$ switching fabric, called a plane, is composed of 2×2 bufferless switching elements arranged in $n=\log_2 N$ stages. The switching fabric considered in this work has no extra stages defined according to [4]. This is indicated by the "0" in $\log_2(N,0,1)$. Each stage contains $N/2$ switching elements, numbered from 0 to $N/2-1$ from top to bottom. Inputs and outputs of the switching fabric are numbered $0,1,\ldots,N-1$ from top to bottom, and stages are numbered $1,2,\ldots,n$ from left to right. Input i of each plane is connected with input i of switching fabric

TABLE I
NOTATION USED THROUGHOUT THE PAPER.

	Symbol	Description
Architecture	N	number of inputs (outputs) in the switching fabric
	p	number of vertically stacked planes
	n	number of stages of switching elements, $n = \log_2 N$
	S_e^s	e-th switching element in stage s
	I_l^s	l-th link between stages s and $s+1$
	$\langle x, y\rangle$	a connection between input link x and output link y
	(X, Y)	a relation, i.e., a set of switching elements and interstage links belonging to connection path of connection $\langle x, y\rangle$
	$L(i)$	a label of element i, i.e., element number expressed in form of a binary number
	m	size of a data structure expressed in the number of elements representing existing connections in a switching fabric
De Bruijn graph	$\mathcal{G}(g, d)$	a De Bruijn graph with d^g vertexes and d^{g+1} arcs
	g	number of symbols in vertex label in a De Bruijn graph
	$\mathbb{Z}_d$	alphabet of symbols used in the De Bruijn graph, $\mathbb{Z}_d = \{0; 1\}$ in this work
	d	number of symbols in $\mathbb{Z}_d$, i.e., $d = \lvert\mathbb{Z}_d\rvert$
	$\mathcal{V}$	set of nodes in a De Bruijn graph
	V^i	vertex in the De Bruijn graph $\mathcal{G}(g, d)$, denoted as $L(V^i) = v_{g-1}^i, \ldots, v_1^i, v_0^i$
	$\mathcal{A}$	set of arcs in a De Bruijn graph
	A^i	an arc in the De Bruijn graph $\mathcal{G}(g, d)$, denoted as $L(A^i) = a_g^i, a_{g-1}^i, \ldots, a_1^i, a_0^i$
TCAM-ALG	B_i	i-th TCAM bank $(0 \leqslant i \leqslant n-1)$
	z_{max}	height of a TCAM bank (number of rows, $z_{max} = N-1$)
	c_{max}	width of a TCAM bank (the number of columns, $c_{max} = 2n$ for B_n and $c_{max} = 2n-2$ for other cases)
	K	searched word, $K = k_{\text{max\_c}-1}, k_{\text{max\_c}-2}, \ldots, k_1, ks_0$
	W^z	a word stored at row z of a TCAM bank, $W^z = w_{c_{max}-1}^z, w_{c_{max}-2}^z, \ldots, w_1^z, w_0^z$
	M^z	a mask stored at row z of a TCAM bank, $M^z = m_{c_{max}-1}^z, m_{c_{max}-2}^z, \ldots, m_1^z, m_0^z$
	RA	a set of signals containing information about existing connections that block a new connection $\langle x, y\rangle$, $ra_z = 1$ indicates that at least one result line of B_i is equal to 1
	RB	a set of signals indicating which word W^z of bank B_i contains label $L((X, Y))$
	C	a matrix binding information about existing connections and used planes, $c_{z,j} = 1$ indicates that connection represented by $L(\langle x, y\rangle)$ and stored in row z is set up through plane j
	P^T	TCAM search result at row z depending on K, W^z and M^z (Eq. (1))

through a DMUX, while a MUX is used to connect output o of each plane with output o of the switching fabric. The switching element e in stage s is denoted by S_e^s in each plane, where $0 \leqslant e \leqslant N/2 - 1$ and $1 \leqslant s \leqslant n$. Let I_l^s denote the interstage link connected to output of switching element S_e^s, where $l = 2e$ or $l = 2e+1$ for link connected to upper or lower output of a switching element, respectively. Since the switching fabric is composed of 2×2 switching elements, the number of links in each stage of interstage links is equal to N. Links are numbered in each stage from 0 to $N-1$ from top to bottom, thus $0 \leq l \leq N-1$.

We define a connection $\langle x, y\rangle$ between input link x and output link y in $\log_2(N, 0, p)$ switching fabric as a set of resources of switching fabric occupied at a given time instance. Resources used by the connection, i.e., a set of switching elements and interstage links, are called a connection path. There is exactly one connection path in $\log_2(N, 0, 1)$ switching fabric between input x and output y. When two connections have to share at least one interstage link, each of them must

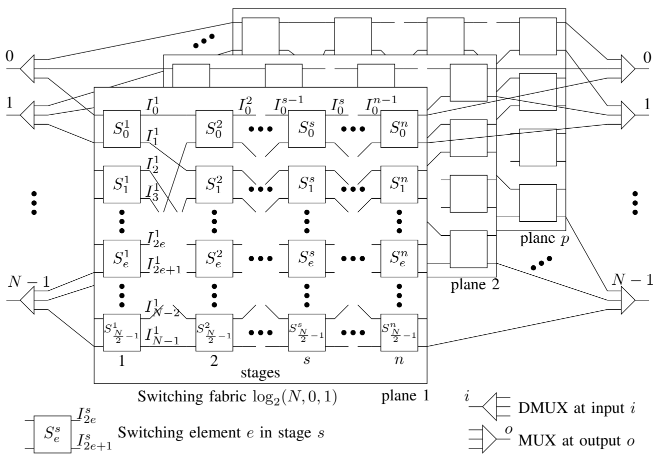


Fig. 2. A $\log_2(N, 0, p)$ switching fabric with a perfect-shuffle topology.

be set up through a different plane. In order to determine the state of the switching fabric, it is sufficient to limit information about each connection to the number of the first stage switching element and the number of the last stage switching element under the assumption that all switching elements are non-blocking. This pair is called a relation (X, Y) between a switching element X at the input stage and a switching element Y at the output stage, where $X = \lfloor x/2 \rfloor$ and $Y = \lfloor y/2 \rfloor$.

B. Control plane

In order to set up a new connection in the data plane, the control plane takes a decision how to realize it. Every time a new connection arrives at an input of a switching node, the control plane uses a control algorithm to find a path to set up this connection. The control algorithm is divided into two parts: searching and forwarding. Searching is one of the most important, complex, and common operations performed in both circuit and packet processing. Forwarding depends on the type of switching fabric and is out of scope of this work.

The most popular structures used in control algorithms, i.e., arrays and linked lists are implemented in RAM. The disadvantage of these structures is complexity of search operations that consumes CPU computing power. Content Addressable Memory (CAM) [8] instead of RAM can be used in order to reduce searching time. CAM is more complex and expensive than RAM, but the address of the searched value is immediately returned upon the search request without using a CPU.

CAM is a hardware component that enables search performance within a single clock cycle. Similarly to RAM, CAM contains conventional semiconductor memory cells. However, CAM is able to simultaneously compare the searched value against all values stored in the CAM. CAM cells are arranged in horizontal words and columns indexed by z and c, respectively, $0 \leqslant z \leqslant z_{max} - 1$ and $0 \leqslant c \leqslant c_{max} - 1$ (see Fig. 3). Each CAM memory cell contains a storage circuit and a comparison circuit. They allow a simultaneous comparison of the searched word $K = k_{c_{max}-1}, \ldots, k_1, k_0$ (delivered to memory cells through search line) with the values stored in each CAM cell (not detailed in Fig. 3 for

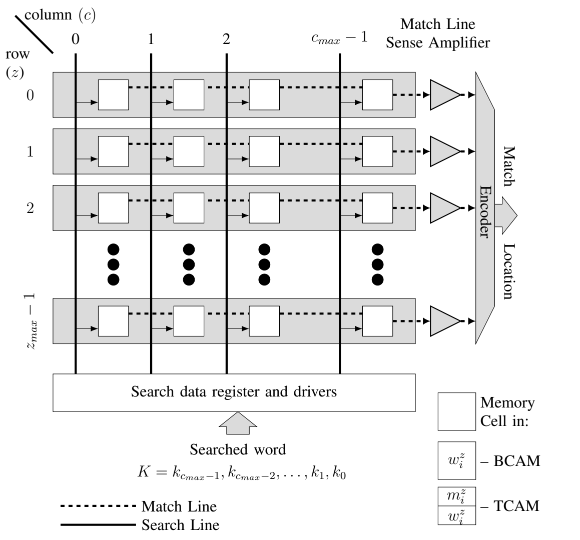


Fig. 3. CAM structure (BCAM and TCAM structures distinguished by different memory cells).

its clarity). Comparison results from particular words are sent to corresponding match lines.

There are two main types of CAM, i.e., Binary Content Addressable Memory (BCAM) and TCAM as indicated in Fig. 3. BCAMs store one of two values (zero and one), whereas TCAMs store one of three values: zero, one, and do not care. The third value allows masked searches if the values are compared with the masked entries stored in TCAM. The comparison in BCAM is simple and requires only stored word $W^z = w^z_{c_{max}-1}, \ldots, w^z_1, w^z_0$, while in case of TCAM, mask $M^z = m^z_{c_{max}-1}, \ldots, m^z_1, m^z_0$ is additionally used to show if a given bit w^z_c is important ($m^z_c = 0$) or not ($m^z_c = 1$).

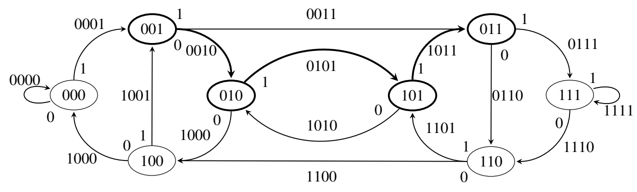
When the search operation is triggered, all bits of the searched word K are compared with corresponding bits of each word in TCAM, and the value of match line z is

$$P^T(K, W^z, M^z) = \prod_{i=0}^{c_{max}-1} \left(\overline{k_i \oplus w^z_i}\right) \vee m^z_i. \qquad (1)$$

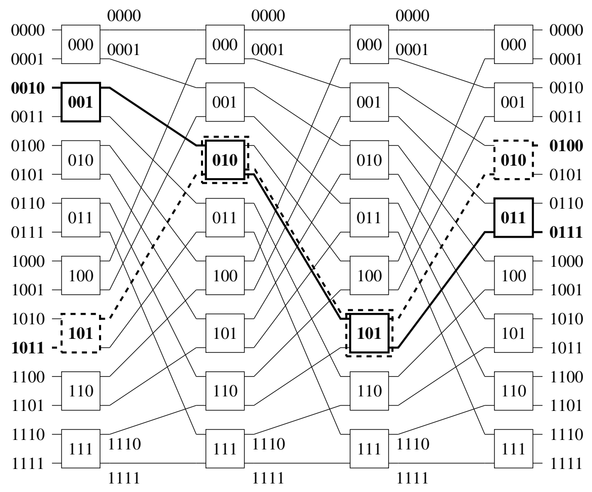
The function $P^T(K, W^z, M^z)$ is used in Section IV to detect if an arriving connection is blocked in a given plane.

III. Switching fabric model with De Bruijn graphs

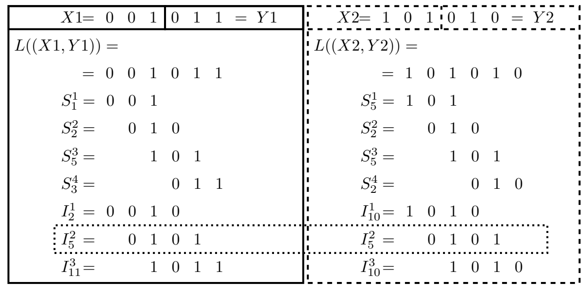
The control plane needs to determine whether an arriving connection can be established through the switching fabric or must be rejected. We use a De Bruijn graph [9] for this purpose. The De Bruijn graph is a useful tool to represent the whole switching fabric because each stage of an interstage link has the same pattern in the perfect-shuffle switching fabric. The labels of De Bruijn graph nodes represent the labels of switching elements. The De Bruijn graph determines (inside one stage of the interstage links) the labels of output switching elements available from input switching elements. Edge labels unequivocally define the relationship, which determines the set of interstage links. The label of interstage link is created by concatenating the label of switching element at stage s and the most significant bit from the label of connected switch in the next stage.



(a) De Bruijn graph $\mathcal{G}(2,3)$.



(b) Connections $\langle 2, 7\rangle$ and $\langle 11, 4\rangle$ in a $\log_2(16, 0, 1)$ switching fabric.



$X1 = 0\ 0\ 1 \mid 0\ 1\ 1 = Y1$	$X2 = 1\ 0\ 1 \mid 0\ 1\ 0 = Y2$
$L((X1, Y1)) =$	$L((X2, Y2)) =$
$= 0\ 0\ 1\ 0\ 1\ 1$	$= 1\ 0\ 1\ 0\ 1\ 0$
$S^1_1 = 0\ 0\ 1$	$S^1_5 = 1\ 0\ 1$
$S^2_2 = 0\ 1\ 0$	$S^2_2 = 0\ 1\ 0$
$S^3_5 = 1\ 0\ 1$	$S^3_5 = 1\ 0\ 1$
$S^4_3 = 0\ 1\ 1$	$S^4_2 = 0\ 1\ 0$
$I^1_2 = 0\ 0\ 1\ 0$	$I^1_{10} = 1\ 0\ 1\ 0$
$I^2_5 = 0\ 1\ 0\ 1$	$I^2_5 = 0\ 1\ 0\ 1$
$I^3_{11} = 1\ 0\ 1\ 1$	$I^3_{10} = 1\ 0\ 1\ 0$

(c) Labels of switching elements and interstage links belonging to relations presented in Fig. 4(b).

Fig. 4. Exploitation of a De Bruijn graph in detection of blocking relations.

A more formal description of the usage of De Bruijn graphs is presented next. It is followed by an example. The De Bruijn graph $\mathcal{G}(g, d)$ is a directed graph composed of a set of vertexes $\mathcal{V} = \{V^i : 0 \leqslant i \leqslant d^g - 1\}$ and a set of arcs $\mathcal{A} = \{A^l : 0 \leqslant l \leqslant d^{g+1} - 1\}$ connecting vertexes in a recirculating d-shuffle pattern, where d and g are non-negative integers. The interconnection function is the same as that in ShuffleNet graphs [10]. In contrast to ShuffleNet graphs, the De Bruijn graph represents only one stage of the $\log_2(N, 0, 1)$ switching fabric [11]. An exemplary De Bruijn graph $\mathcal{G}(2, 3)$ is presented in Fig. 4(a). Each vertex and each arc in $\mathcal{G}(g, d)$ are labeled. Every label contains respectively g and $g+1$ symbols from $\mathbb{Z}_d$ alphabet. Since we consider a switching fabric composed of 2×2 switching elements, $\mathbb{Z}_d = \{0, 1\}$. Let $L(V^i) = v^i_{g-1}, \ldots, v^i_1, v^i_0$ and $L(A^l) = a^l_g, a^l_{g-1}, \ldots, a^l_1, a^l_0$ ($v^i_x, a^l_x \in \mathbb{Z}_d$) denote labels of vertex V^i and arc A^l in the De Bruijn graph, respectively. The vertex V^i is connected with the vertex V^j by an arc A^l if

$$L(A^l) = v^i_{g-1}, \dots, v^i_1, v^i_0, v^j_0 = v^i_{g-1}, v^j_{g-1}, \dots, v^j_1, v^j_0, \quad (2)$$

i.e., each arc is labeled by the most significant symbol of the label of the originating vertex, followed by the label of the destination vertex. Moreover, since each output arc attached to a given vertex is determined by a symbol from $\mathbb{Z}_d$, the arc label is constituted by a concatenation of the label of originating vertex and the arc output symbol (see the arc labeled "0001" as a concatenation of node label "000" and arc output symbol "1" in Fig. 4(a)). The label of destination vertex is equal to g less significant symbols of the label of incoming arc. Labels of vertexes and arcs determine respectively labels of each switching element S^s_e and each interstage link I^s_l for a given stage s of a switching fabric, i.e., $L(S^s_e) = L(V^e)$ and $L(I^s_l) = L(A^l)$. Let us also define a label of relation (X, Y) as a concatenation of labels of S^1_X and S^n_Y:

$$L((X, Y)) = x_{n-1} \dots x_2, x_1, y_{n-1} \dots y_2, y_1, \quad (3)$$

where $x_{n-1} \dots x_2, x_1, x_0$ and $y_{n-1} \dots y_2, y_1, y_0$ are binary representations of an input link x and an output link y, respectively. The labels of switching elements and interstage links belonging to the connection path $\langle x, y \rangle$ at each stage of the switching fabric can be extracted from $L((X, Y))$. The label of relation (X, Y) on the first $n-1$ (the most significant) bits contains the label of S^1_X. The next symbol is the most significant bit from the label of S^n_Y. These $n - 1 + 1 = n$ bits represent the label of I^1_l. According to Eq. (2), this label contains the label of S^2_e on the $n - 1$ least significant bits. Thus, starting from bit $2n - 3$ of $L((X, Y))$, the next $n - 1$ bits contain label of S^2_e. When we add the $(n - 4)$-th bit to label of S^2_e, we obtain the label of I^2_l. The presented operation is performed for each stage of interstage links, i.e., $n-1$ times.

As an example let us consider connection $\langle 2, 7 \rangle$, and corresponding relation $(1, 3)$, in $\log_2(16, 0, 1)$ switching fabric indicated with a bold solid line in Fig. 4(b). The relation $(1, 3)$ is represented in the De Bruijn graph (Fig. 4(a)) by vertexes and arcs with the same labels as the labels of switches and interstage links belonging to $(1, 3)$ in Fig. 4(b) (also indicated with a bold solid line). The symbol of outgoing link in each vertex indicates if a connection is set up using upper (0) or lower (1) switch output. We can see that a label composed of symbols of outgoing links is the same as the label of S^4_2, i.e.,010. It corresponds to self-routing properties of $\log_2(N, 0, 1)$ switching fabric, i.e., the ability to take decision how to route received data based only on the label of destination switching element. In Fig. 4(b) an additional connection $\langle 11, 4 \rangle$ (represented by relation $(5, 2)$) is indicated by bold dashed line. Labels of all elements belonging to connection path of relation $(1, 3)$ and $(5, 2)$ are presented in Fig. 4(c). Labels of all switching elements and interstage links are determined based on $L((1, 3))$ and $L((5, 2))$. As we can see both relations share three elements: S^2_2, I^2_5, S^3_5. Since the shared elements contain an interstage link (marked in Fig. 4(c) by doted rectangle), these connections must not be set up through the same plane.

IV. Proposed Node Control Algorithm TCAM-ALG

A control algorithm is required to realize an arriving connection over a switching fabric presented in the previous section. The control algorithms for switching fabric can be divided into two major groups, i.e., serving unicast connections and serving multicast connections [4]. Both of them can be divided into three sub-groups: path searching algorithms, rearrangement algorithms, and repacking algorithms. In order to conduct a fair comparison of existing and proposed algorithms, in this paper we focus only on the path searching algorithms for unicast connections. These algorithms choose a path (equivalent to choosing a plane in our case) to realize a connection or reject it in case no path is available. Several path searching algorithms for unicast connections have been proposed in the literature, e.g., random, sequential, packing, matrix algorithms [4, 12]. Searching algorithms should have low complexity in order to be energy-efficient. Taking into account this requirement, we consider the following energy-unaware algorithms in our study:

- Sequential – a plane for arriving connection is sequentially searched for starting from plane $i = 0$ up to plane $p - 1$ [4];
- Packing (Beneš) – the arriving connection is established through the most loaded but available plane [4];
- Matrix – the arriving connection is established through such a plane in which this connection blocks the fewest possible future connections [12].

We use the model of switching fabric described in Section III to check if a plane of $\log_2(N, 0, p)$ switching fabric with the perfect-shuffle topology is available for the arriving connection. Recall that the label of relation $L((X, Y))$ is composed of $2n - 2$ symbols of alphabet $\mathbb{Z}_d$ (see Eq. (3)). Two relations block each other if at least n sequential symbols in their labels are equal. Each relation label may be presented as $n - 1$ subsequences composed of n symbols as presented in Fig. 4(c) using labels of interstage links I^s_l. Hence, to find every possible blocking configuration, the comparison must be repeated $n - 1$ times. This operation is realized simultaneously in TCAM banks (sets of TCAM cells) that represent a particular stage of interstage links. Based on this simultaneous operation, we propose a control algorithm with the idea similar to the sequential algorithm, but realized with TCAM. This algorithm, called TCAM-ALG, is divided into two parts. The first part (Alg. 1) is used to select the plane for the new connection. It is triggered by the connection arrival at an input link. The second part (Alg. 2) starts immediately after an existing connection is terminated in order to update information about the state of the switching fabric. All steps of the *for* loops in Algs. 1 and 2 are performed simultaneously in hardware. The *for* loops are shown in Algs. 1 and 2 for clarity of presentation.

The TCAM-ALG is implemented in TCAM that is divided into banks B_i, $1 \leq i \leq n$ (see Fig. 5). Decomposition of TCAM into banks is usually used to reduce its power consumption by deactivating unused banks [13]. In our work,

Algorithm 1 Connection establishment in TCAM-ALG.

```
Input: $\langle x,y\rangle$, $B_1,B_2,\dots,B_{n-1},B_n$, $C$
Output: Plane $k$ ($0\le k\le p$) used to set up new connection
    $\langle x,y\rangle$, updated $B_1,B_2,\dots,B_{n-1},B_n$, $C$
 1: Determine $L(\langle x,y\rangle)$, Eq. (5);
 2: Determine $L((X,Y))$, Eq. (3);
 3: Determine $RA$ and $RB$, Eqs. (6) and (7);
 4: for $(i=0; i\leqslant n-1; i{+}{+})$ do
 5:     Determine $f_i$, Eq. (8);
 6: end for
 7: Choose the first $f_i=0$, where $k=i+1$ indicates the
    plane chosen to set up the new connection; If all $f_i=1$
    there is no available plane for $\langle x,y\rangle$, a new connection is
    rejected, and the algorithm terminates;
 8: for $(z=0; z\le N-1; z{+}{+})$ do
 9:     Determine $e_z$, Eq. (9);
10: end for
11: Choose the first $e_z=0$, where $z$ indicates the first empty
    $W^z$ in TCAM banks;
12: for $(i=1; i\leqslant n-1; i++)$ do
13:     Write $W^z=L((X,Y))$ into $B_i$;
14: end for
15: Write word $U^z=L(\langle x,y\rangle)$ to $B_n$;
16: $c_{z,k}=1$;
```

Algorithm 2 Connection release in TCAM-ALG.

```
Input: $\langle x,y\rangle$, $B_1,B_2,\dots,B_{n-1},B_n$, $C$
Output: Updated $B_1,B_2,\dots,B_{n-1},B_n$, $C$
 1: Determine $L(\langle x,y\rangle)$, Eq. (5);
 2: Determine $RA$ and $RB$, Eqs. (7) and (7);
 3: Based on $RB$ determine $z: e_z=1$, Eq. (9);
 4: for $(i=0; i\leqslant p-1; i++)$ do
 5:     $c_{z,i}=0$;
 6: end for
```

this technique is used to separate search of blocking relation in particular stages of interstage links, while power reduction is achieved with a control algorithm (TCAMs allow parallel lookups of blocking paths with time complexity of O(1) [14]). Each B_i for $1\leqslant i\leqslant n-1$ is composed of $N\times(2n-2)$ TCAM cells and contains N words W^z masked by M^i. The words W^z in banks B_i, $1\leqslant i\leqslant n-1$, store relation labels of existing connections. The mask M^i in bank B_i is used to indicate important bits, i.e., the bits on positions in the relation label $L((X,Y))$ corresponding to stage i of interstage links. In a given bank all masks m_j^j are the same and are given by:

$$m_j^i=\begin{cases}0 & \text{if } j\leqslant i+1\le j+n \text{ and } 0\leqslant i\leqslant n-1,\\ 0 & \text{if } i=n,\\ 1 & \text{for other cases.}\end{cases} \quad (4)$$

The bank B_n ties connection $\langle x,y\rangle$ with particular row z, $0\le z\le N-1$ in all banks B_i, where $1\leqslant i\leqslant n-1$. The words U^z and masks M^i in B_n are composed of $2n$ bits. The bank B_n contains labels of connections existing in the $\log_2(N,0,p)$ switching fabric. These labels, denoted as

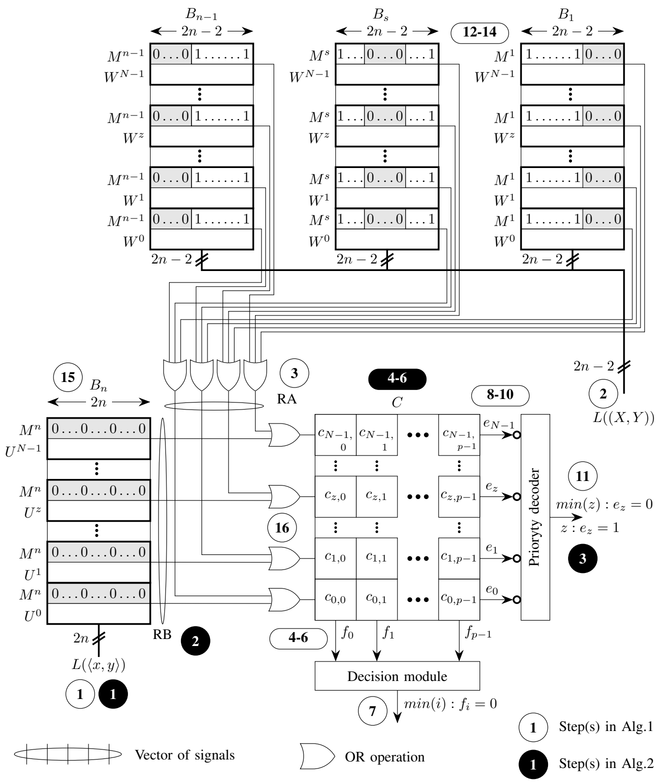


Fig. 5. Division of TCAM in TCAM-ALG.

$L(\langle x,y\rangle)$, are created by concatenating binary representations of an input link number x and an output link number y, i.e.,

$$L(\langle x,y\rangle)=x_{n-1}\dots x_2,x_1,x_0,y_{n-1}\dots y_2,y_1,y_0 \quad (5)$$

as the word U^z masked by M^n with all bits equal to 0. As presented in Fig. 3, each word W^z in TCAM has its own match line with state given by Eq. (1). Signals (bits) from particular result lines are logically summed to obtain a set of lines RA which contains information about existing connections that block a new connection. The existence of element ra_z indicates that at least one B_i, $1\leqslant i\leqslant n-1$ on result line z is equal to 1. The set RA is defined as follows:

$$RA=\Big\{ra_z: ra_z=(z+1)\cdot\bigcup_{i=1}^{n-1}P^T(L((X,Y)),W^z,M^i),\; ra_z>0, 0\leqslant z\leqslant N-1\Big\}. \quad (6)$$

Multiplication by $z+1$ (rather than z) is performed only to solve the ambiguity caused by multiplication indexes (zero-based) and values returned by Eq. (1). Similarly, we define a set of lines RB used to indicate which W^z at bank B_i represents $\langle x,y\rangle$. This set is defined as

$$RB=\big\{rb_z: rb_z=(z+1)\cdot P^T(L(\langle x,y\rangle),U^z,M^n),\; rb_z>0, 0\leqslant z\leqslant N-1\big\}. \quad (7)$$

Both sets RA and RB form an address of matrix C binding information about existing connections and occupied planes. The matrix C is composed of N rows and p columns. The value $c_{z,j}$ equal to 1 means that relation represented by word W^z in row z in every TCAM bank is set up through plane $j+1$. The matrix C has two sets of result lines, i.e., horizontal and vertical lines. The vertical result lines are connected to a decision module through p lines denoted as f_i, $0 \leqslant i \leqslant p-1$. The state of a particular f_i depends on RA and on the content of C. It is calculated as follows:

$$f_i = \bigcup\nolimits_{\forall j \in RA} c_{j-1,i} \tag{8}$$

If $f_i = 1$, connection $\langle x, y\rangle$ is blocked in plane $i+1$.

Signals e_z for $0 \leqslant z \leqslant N-1$ indicate the first available W^z that will be used by a new connection in case of Alg. 1 or they indicate which W^z stores $L((X,Y))$ to be released (in case of Alg. 2). The state of particular e_z is given by:

$$e_z = \begin{cases} \bigcup\nolimits_{\forall j \in RB} c_{z,j-1} & \text{for } j > 0 \text{ in case of Alg.1,} \\ \bigcup\nolimits_{j=0}^{p-1} c_{z,j} & \text{in case of Alg.2,} \end{cases} \tag{9}$$

Connection establishment: When a new connection request $\langle x, y\rangle$ arrives at the switching node input interface, the control module has to find a plane to set up this connection (Alg. 1). The labels $L(\langle x, y\rangle)$ and $L((X,Y))$ are determined respectively in Steps 1 and 2 of Alg. 1. Next, bits of $L((X,Y))$ are sent to B_i ($1 \leqslant i \leqslant n-1$) as searched words. The results of matching operation in each B_i ($1 \leqslant i \leqslant n-1$) are sent to corresponding match lines. The set RA is determined based on their states (Step 3). Signal $ra_z = 1$ indicates that at least one B_i on result line z is equal 1, i.e., a connection represented by W^z in plane i blocks the arriving (new) connection. We determine values of all signals f_i based on the set RA and matrix C, and we choose the line $f_i = 0$ with the smallest i (Step 7), i.e., the plane $k = i+1$ will be used to set up connection $\langle x, y\rangle$. Next, the state of switching fabric must be updated. In Steps 8–10, available positions (not used by existing connections) in TCAM banks are determined, and the first available position (with the smallest index z) is chosen (Step 11). Next, $L((X,Y))$ and $L(\langle x, y\rangle)$ are written (Steps 12-14 and Step 15, respectively) to appropriate TCAM banks at the position chosen in Step 11. The matrix C is updated in Step 16.

Connection release: Alg. 2 is triggered when an established connection terminates. In Step 1, $L(\langle x, y\rangle)$ is determined. This label is used in Step 2 as the searched word for B_n. As a result of the searching, we obtain the set RB containing exactly one element equal to 1, which corresponds to position at which released connection is stored. In Step 3, we determine, using a priority decoder, the number z that corresponds to the position used in TCAM banks by $\langle x, y\rangle$. The Steps 4–6 of Alg. 2 remove this connection from matrix C.

V. Evaluation

A. Scenarios and metrics

We evaluate TCAM-ALG and the three alternative algorithms in a simulative way using two evaluation metrics. First, we calculate energy consumed by the control module to process all connections over a day regarding their establishment and release. We use the energy model from [15, 16] and the values of energy quanta for a particular CPU operation (e.g., division) from [17]. A detailed description of the considered CPU operations are omitted due to space constraints.

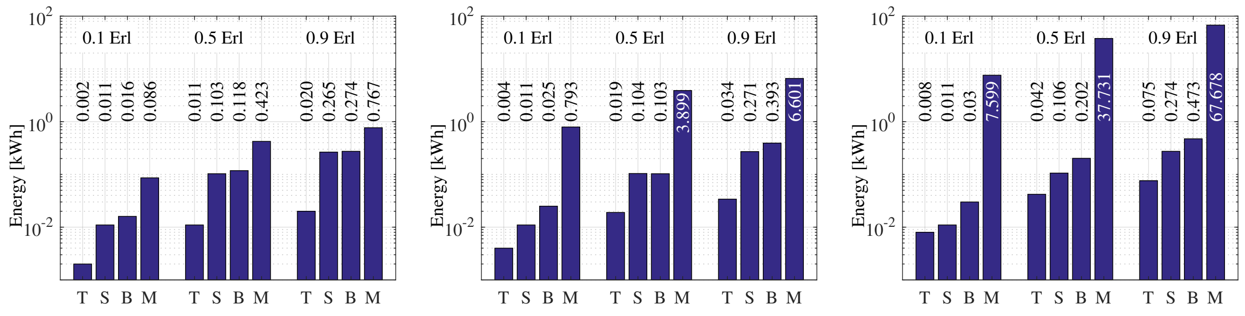
As shown above, the proposed algorithm determines connection path without using a CPU. Thus, the energy consumed by the control module is equal to energy dissipated by TCAM (varying from 0.7 to 17.2 fJ/bit/search depending on the match line architecture [6]). Thus, the energy required by a given TCAM chip depends only on the memory size and the number of searches requested in a given time period. Energy consumption of the most popular TCAM, i.e., the NOR TCAM is estimated to 5.3 fJ/bit/search, and used in this evaluation.

The second metric used to compare TCAM-ALG with the existing algorithms is the cost of used memory expressed in the number of required transistors. We assume that the Beneš, sequential, and matrix algorithms are executed in a control module using Synchronous Dynamic RAM (SDRAM). SDRAM requires only one transistor per bit while a bit of TCAM corresponds to 6–18 transistors depending on the TCAM architecture. We assume that a single cell of memory (with a mask) contains 8 transistors for NOR TCAM [6, 18].

B. Results

Energy consumed by a control module (running different algorithms) to find paths for all arriving connections over 24 hours for switching fabric size $N = 8, 16, 32$ and for different offered traffic is presented in Fig. 6. Capacity of each interface is equal to 10 Gbps. We use Poisson traffic model with mean packet length 454 Bytes and offered traffic varied between 0.1 and 0.9 Erl. Energy consumed by the control module working under TCAM-ALG, Beneš, and sequential algorithms slightly grows with increasing offered traffic and N (differences lower than 0.5 kWh/day). The increase of energy consumption in function of N is exponential in case of the matrix algorithm. Furthermore, the amount of energy consumed in case of TCAM-ALG is lower by an order of magnitude than in case of the Beneš and sequential algorithms (the ratio of energy consumed in case of TCAM-ALG to other algorithms varying in range 0.07–0.36), and by a few orders of magnitude than the matrix algorithm (0.001–0.023). The poor performance of the matrix algorithm is caused by the fact that the size of matrices used in this algorithm increases exponentially, and with the same rate increases the number of required updates in the matrices. Moreover, for a given N, the energy consumption increases faster for the three alternative algorithms than for TCAM-ALG with increasing offered traffic.

Fig. 7 presents costs of memory used to store state of the switching fabric expressed in the number of used transistors. We can observe that the proposed algorithm requires more transistors than the Beneš and sequential algorithms. Although more expensive in terms of transistors, TCAM-ALG does not use any CPU. The cost of CPU is difficult to express in terms of used transistors, because the CPU may be used to realize tasks other than path searching. A specific cost metric



(a) $N = 8$ (b) $N = 16$ (c) $N = 32$

Fig. 6. Energy [kWh] consumed by a control module over a day when using different control algorithms for varied switching fabric size N and different offered traffic [Erl] (algorithms: T – TCAM-ALG, S – sequential, B – Beneš, M – matrix).

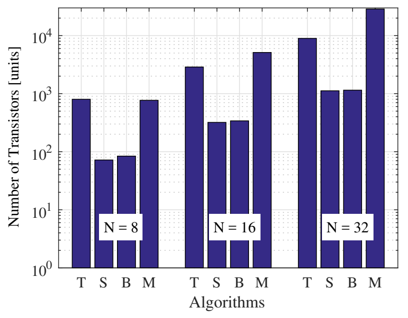


Fig. 7. Memory costs (algorithms: T – TCAM-ALG, S – sequential, B – Beneš, M – matrix).

to account for both transistors and CPU usage is left for future work. Furthermore, we point out that the proposed TCAM-ALG requires comparable or lower number of transistors than the matrix algorithm.

VI. Conclusion

We propose a control module that uses De Bruijn graphs, TCAM, and no CPU to determine paths for connections arriving at a network node. Daily energy consumption of the control module running the proposed TCAM-ALG is reduced by up to 100 times in comparison to three alternative algorithms known from the literature. On the other hand, two alternative algorithms, i.e., Beneš and sequential algorithm require memory with fewer transistors than TCAM-ALG. As future work, we are going to check whether the high usage of transistors required by TCAM in TCAM-ALG is compensated in terms of Operational Expenditure by no need of a CPU. We are also working on further optimization of the proposed algorithm in order to limit the size of used TCAMs.

Acknowledgment

The work of Filip Idzikowski was supported by the National Science Centre, Poland (decision DEC-2014/12/S/ST7/00415). The work of Mariusz Żal was supported by the funds from the Polish Ministry of Science and Higher Education for the year 2017 under Grant 08/82/DSPB/8221.

References

[1] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future Internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 223–244, Second Quarter 2011.

[2] A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier, "A survey of green networking research," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 3–20, First Quarter 2012.

[3] F. Idzikowski, L. Chiaraviglio, A. Cianfrani, J. López Vizcaíno, M. Polverini, and Y. Ye, "A survey on energy-aware design and operation of core networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1453–1499, Second Quarter 2016.

[4] W. Kabaciński, *Nonblocking Electronic and Photonic Switching Fabrics*. Springer, 2005.

[5] H. Huang, S. Guo, J. Wu, and J. Li, "Green DataPath for TCAM-based Software-Defined Networks," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 194–201, Nov. 2016.

[6] S. H. Yang, Y. J. Huang, and J. F. Li, "A low-power ternary content addressable memory with Pai-Sigma matchlines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 10, pp. 1909–1913, Oct. 2012.

[7] C.-T. Lea, "Multi-log$_2N$ networks and their applications in high-speed electronic and photonic switching systems," *IEEE Transactions on Communications*, vol. 38, no. 10, pp. 1749–1740, Oct. 1990.

[8] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, March 2006.

[9] N. G. de Bruijn, "A Combinatorial Problem," *Koninklijke Nederlandsche Akademie Van Wetenschappen*, vol. 49, no. 6, pp. 758–764, June 1946.

[10] M. G. Hluchyj and M. J. Karol, "ShuffleNet: an application of generalized perfect shuffles to multihop lightwave networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 9, no. 10, pp. 1386–1397, Oct. 1991.

[11] F. Tekiner, Z. Ghassemlooy, S. Al-Khayatt, and M. Thompson, "Implementation and evaluation of ShuffleNet, Gemnet and De Bruijn graph logical network topologies," in *Proc. PDCN, Innsbruck, Austria*, Feb. 2004.

[12] W. Kabaciński and M. Michalski, "The routing algorithm and wide-sense nonblocking conditions for multiplane baseline switching networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 35–44, Dec. 2006.

[13] B. D. Yang, "Low-power effective memory-size expanded TCAM using data-relocation scheme," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 10, pp. 2441–2450, Oct. 2015.

[14] Y. Nozaki, N. Shenoy, and A. Gupta, *Power Usage Efficiency with a Modular Routing Protocol*. Springer International Publishing, 2016.

[15] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 4, pp. 797–804, Apr. 2008.

[16] J. T. Russell and M. F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in *Proc. ICCD, Washington, USA*, Oct. 1998.

[17] S. Nikolaidis, N. Kavvadias, P. Neofotistos, K. Kosmatopoulos, T. Laopoulos, and L. Bisdounis, "Instrumentation set-up for instruction level power modeling," in *PATMOS*, Jan. 2002.

[18] P. J. Ashenden, *Digital Design (VHDL): An Embedded Systems Approach Using VHDL*. Elsevier Science, 2007.