Method of Filtering Tables Design for Ultra-fast Switching for Systems with Limited Resources

Sergey Makov

***Abstract*—the paper discusses designing of the frame filtering tables in distributed computing or telecommunication systems. The proposed method of filtering tables design can reduce the time of frame processing by network bridges and switches and provide a low probability of filtering table overflowing. We have studied the new method and determined optimal distributing of memory amount for table allocating.**

Also we discuss the energy efficient of devices in dependence on filtering table design. Hash table is a common approach to build associative arrays, database indexes and various kinds of program-defined caches. Our approach allows to design ASIC to perform function of fast search in associative arrays. It leads to significant decreasing power consumption. Moreover, hashing techniques are suffered from large probability of collision in the case of hash size acceptable for mobile devices. This makes it necessary to perform additional energy-inefficient memory access operations to resolve these collisions. We propose hashing technique with lower probability of collision for the hash of the same size. We show that unlike existing collision free approaches our hashing method has a much broader area of applicability. To support these claims both theoretical and experimental studies are presented. Experimental comparison with existing approaches has shown significant improvement of energy-efficiency for common applications.

***Index Terms*—hash table; hash collision; energy efficiency; lookup table; associated table; frame forwarding; packet switching; packet routing; IoT.**

I. Introduction

Amount of mobile devices and IoT devices recent time is dramatic increasing. It is leading to load increasing on the network switches and routers. Mobile devices and IoT devices are using radio for data transfer. The primary function of switches and routers is isolating of local traffic. Absence of this isolation will lead to decreasing of bandwidth of data channel due to sharing the channel between devices and collisions of packets. That's why the problem of efficient switching and local traffic isolation is still relevant.

Another modern trend is using the mobile devices as network switches and routers. It can be 4G – WiFi routers on smartphones or switch for mesh-network of IoT with autonomic power supply. Such devices should work fully autonomously in autonomous mode using portative power supplies only. It makes their energy efficiency an extremely important problem [1].

There are few ways to improve energy efficiency. One of them is to improve microelectronics technology. Decreased electrical capacity of the gate leads to reduction of power consumption per each gate in digital circuit. It can be achieved by reduction of gate size. [2] Another way is dynamical control of voltage supply and clock frequency of digital circuits according to performed tasks. This approach suggests compliance of operating systems and applications for mobile devices to given requirements [3]. The main mechanism is to switch the mobile device in "sleep mode" that have to be supported by operating system. [4] All these ways are universal and independent on algorithms used within applications and operating system functions.

In order to ensure high energy-efficiency, algorithms targeted to mobile and IoT devices should be developed with minimal memory and computational requirements. This is general requirement for all algorithms. Other requirement is speed. In many cases these requirements cannot be achieved simultaneously.

One of the common types of algorithms is ones based on lookup-tables. Some applications require determining if some data point belong to one or another group based on some criteria. For instance, it can be classification of pictures according presence of particular person or smiling faces in it. Another important example is separation of descriptors according to some features.

In some applications there is a need to have ability to modify behavior and update its lookup-tables when objects transit from one class to other or when information about object's class becomes irrelevant. Important example is the routing table in mesh network of mobile devices. When some device appears or disappears in the network area or move from one location to another. Another example is switching tables in network switches.

These tasks are usually being solved by self-learning lookup-tables. That's why development of methods for lookup-table design and efficient algorithms to update or search information within the table is an actual goal.

Some of application dependent of lookup-tables are supposed to work in real-time. In such situations the time to update data or to perform a search within it is limited. Quite often this time may be about few microseconds. This is the case for packet switching or routing with use of self-learned lookup-tables. All operations in this type of tables should be performed in time interval between two packets as shown on figure 1 for Ethernet. In case of on-the-fly processing or cut-through switching the time to search data in the table for correct switching or routing could be limited by nanoseconds.

Sergey Makov is with the Radio and electro-technical Department, Don State Technical University of Rostov-on-Don, Shakhty, Rostov reg., Russia, (e-mail: makovs@rambler.ru).

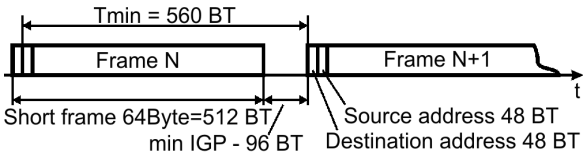

Fig. 1. Time to processing algorithm

For 10GBit/s systems this time is equals to 56ns. It is hard to make a decision during this time using low cost microcontrollers or processors. Due to that fact the ASICs are using additional hardware for computing. [5]

In the case of cut-through switching the search time should be as short as possible to minimize frame buffer size and latency. Decision time interval can be about few nanoseconds even for 1GBit/s systems.

The most efficient way of lookup-tables design is hashing tables [6]. However, this approach has a disadvantage – there is possibility for several different search keys to have equal hash. This situation is called "collision". There are many was to resolve collisions in hashing tables. All of them take additional time and computational resources. Not resolved collision is the cause of wrong search result. In case of classification task it means false negative or false positive results.

In this paper we discuss methods of hashing lookup-table design for applications in which is able to have non-zero probability of false positive or false negative results of search. We show that convenient hashing tables design methods have lacks and propose method to improve hashing table efficiency. Also we propose the method to find the "ideal" hashing strategy for applications in which searching results should be exact.

II. HASHING EFFICIENCY ASSESSMENT

The method of blocks is convenient method for hashing tables collision resolving. According to this method for each hash value several cells is allocated to store search key. In the case of collision the search continues in next cell until the match is found or an empty cell is reached. We have shown that this method does not eliminate the probability of collision, but rather provides a reduction to an acceptable value. To assess the efficiency of table design method we compare amount of memory to store the lookup-table and the probability of non-resolved collision.

The collision probability is the ratio of number of all possible combinations of searching keys that give us collision Q and the number of all possible combinations of searching keys W:

$$P = \frac{Q}{W} \tag{1}$$

To find the collision probability let's make following definitions:

$r \cdot l$ – is the number of all possible searching keys;

r – the number of possible hash values;

m – the number of existing at the moment searching keys;

k – the number of cells in the table for one hash value $k \le l$.

Below we use the following denotes:

$|X|$ – the number of elements of a finite set X;

$[t]$ – the integer part of real number t;

$C_n^m = \frac{n!}{m!(n-m)!}$ – the number of m-combinations of a set of n elements, for $m \ge n \ge 0$.

We can think of hashing process as the process of dividing of the set of all search keys A to subsets $A_1, A_2, ..., A_r$ with l elements in each. The hash value for each search key in one subset is the same value $h(x_1) = h(x_2) = ... = h(x_l)$ for $x \in A_i$. $A = \bigcup_{i=1}^{r} A_i$.

The set B is a subset of A and it consists of all existing searching keys. Let's call the set B "$\{A_i\}_{i=1}^{r}, m, k$–allowable" when for each $i = 1,2,...,r$ the set A_i is contains no more than k elements from set B. The number of all allowable sets we denote as λ_{rl}^{mk}. To determinate collision probability we should find $Q = W - \lambda_{rl}^{mk}$ and use equation (1). Here

$$W = \frac{(rl)!}{m!n!(rl-m-n)!}. \tag{2}$$

Let's derive an expression to count all allowable sets $\lambda_{rl}^{mk} = \left|\left\{B \subset A : |B| = m; |B \cap A_i| \le k; i = 1,2,...,r\right\}\right|$. It is clear that λ_{rl}^{mk} is positive if and only if

$$m \le rk \tag{3}$$

Suppose that inequality (3) is satisfied. In this case if $k = 0$ then $m = 0$ and the only one allowable set is $B = \varnothing$. It means $\lambda_{rl}^{00} = 1 \forall r, l \ge 0$.

Let's consider the case when $k \ge 1$. For every allowable set B we will take $T_B = \left\{1 \le i \le r : |B \cap A_i| = k\right\}$, $\overline{T_B} = \{1,2,...,r\} \setminus T_B$. Let's find the range of values $s = |T_B|$. Any allowable set B satisfy to conditions:

$$\left|B \cap \left(\bigcup_{i \in T_B} A_i\right)\right| \le m, \quad \left|B \cap \left(\bigcup_{i \in \overline{T}_B} A_i\right)\right| \le (k-1)|\overline{T}_B|.$$

These conditions may be rewritten as system of inequality:

$$\begin{cases} ks \le m \\ m - ks \le (k-1)(r-s) \end{cases} \Leftrightarrow m - r(k-1) \le s \le \frac{m}{k}$$

or

$$\max(0; m - r(k-1)) \le s \le \left[\frac{m}{k}\right] \tag{4}$$

Opposite is also true. For each integer s from band (4) may be found allowable set B satisfied the condition: $|T_B| = s$. To construct such a set we need to perform next procedures:

take a s-elements subset T_B of set $\{1,2,...,r\}$;

if T_B is not empty, for each $i \in T_B$ take k-elements set B_i

from A_i ;

take "$\{A_i\}_{i\in\overline{T}_B}, m-ks, k-1$ - allowable" set D from $(r-s)l$ -elements set $A' = \bigcup\limits_{i\in\overline{T}_B} A_i$;

suppose $B = \left(\bigcup\limits_{i\in T_B} B_i\right) \bigcup D$.

The first procedure can be accomplished in C_r^s ways. The second one in $\left(C_l^k\right)^s$ ways. Third – $\lambda_{r-s,l}^{m-ks,k-1}$ ways. According to the principle of multiplication we have the number of "$\{A_i\}_{i=1}^{r}, m, k$ - allowable" sets:

$$\lambda_{rl}^{mk} = \sum_{s=\max(0;m-r(k-1))}^{\left[\frac{m}{k}\right]} C_r^s \left(C_l^k\right)^s \lambda_{r-s,l}^{m-ks,k-1} \qquad (5)$$

From (3) and (4) it is clear that all values denoted as λ_{il}^{jp} are positive so we can use (5) for finding λ_{il}^{jp} with appropriate values of i, j, p. Thus we have a recurrent expression to count all allowable sets.

Let's derive expressions for collision probability for lookup-tables designed by convenient method.

Using (1), (2) and (5) we will have:

For k=1

$$P = 1 - \prod_{v=1}^{m} \frac{(r-m+v)l}{rl-m+v}.$$

For k=2 we have:

$$P = 1 - \sum_{s=\max(0;m-r)}^{\left[\frac{m}{2}\right]} \left(\prod_{v=1}^{m} \frac{l}{(rl-m+v)} \cdot \prod_{w=1}^{2s} (m-2s+w) \times \right.$$

$$\left. \times \prod_{t=1}^{m-s} (r-(m-s)+t) \cdot \prod_{u=1}^{s} \frac{l-1}{2lu} \right).$$

For k=4:

$$P = 1 - \sum_{s=0}^{\left[\frac{m}{4}\right]} \sum_{p=0}^{\left[\frac{m-4s}{3}\right]} \sum_{q=0}^{\left[\frac{m-4s-3p}{2}\right]} \left\{ \prod_{v=1}^{m} \frac{l}{rl-m-v} \times \right.$$

$$\times \prod_{w}^{4s+3p+2q} (m-4s-3p-2q+w) \times$$

$$\times \prod_{t=1}^{m-3s-2p-q} (r-(m-3s-2p-q)+t) \prod_{u=1}^{s} \frac{(l-1)(l-2)(l-3)}{24l^3u} \times$$

$$\left. \times \prod_{y=1}^{p} \frac{(l-1)(l-2)}{6l^2y} \cdot \prod_{j=1}^{q} \frac{(l-1)}{2lj} \right\}.$$

As you can see, the expression becomes more difficult with the growth of the number of cells in a block.

For example for the Ethernet MAC-addresses $r \cdot l$ is equal to 2^{48}. For 10-bit hash r = 1024. Fig. 2 shows dependence of the collision probability from the number of nodes in net m of blocks k=1, k=2 and k=4. For this example the required amount of memory for lookup-table is 256Kbits.

In the system proposed by the McNeil patent [7] with a memory amount of 8 Mbit, a 15-bit hash was used. We have calculated the probability of collisions for the system. It was approximately 10^{-7} for 1000 network's nodes.

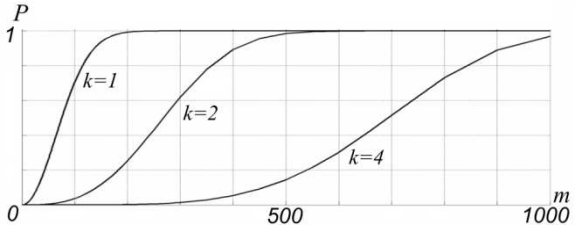

Fig. 2. Collision probability for convenient method

Thus even for method of blocks the collision probability do not reach zero. We can provide acceptable level of collision probability only by use of different methods to reduce this probability, such as parallel hashing or adaptive hashing.

III. PROPOSED METHOD OF TABLE DESIGN

Any classification database can be represented as set of labels which associate objects to particular class. For example, in case of use of hashing table as filtering table within network switch, we can split filtering table to several ones according to the number of switch ports as shown at figure 3.

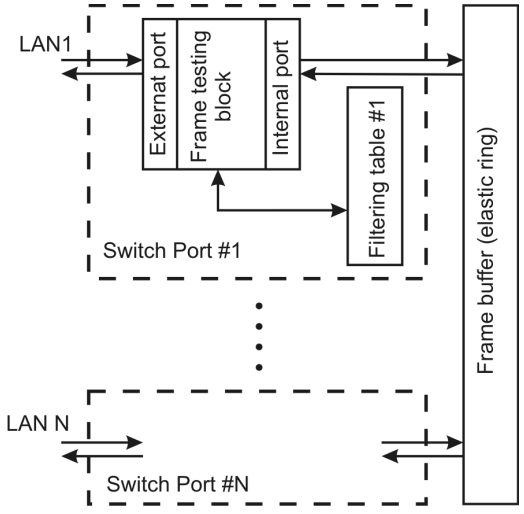

Fig. 3. Switch with splitted filtering table

We propose to exclude the storage of the searching keys in the table. According to proposed method, only the information on the object's class should be stored in the table. We propose to store two flags in lookup-table. One of the flags has non-zero value when an object belongs to a given class. In other words, searching key is belonging to the set of existing keys. The other flag is non-zero if object is not of this class. Thus the amount of required memory to store the table will decrease. But we will not be able to use a mechanism of

collision resolving.

The collision in this case of lookup-table design takes the place only if both flags have non-zero value for the same hash.

So we should estimate the collision probability for proposed method. To solve this problem we use previous definitions with some additions:

m – the number of existing at the moment searching keys belonged to the class;

n – the number of existing at the moment searching keys not belonged to the class.

Let's suppose set B contains searching keys belonged to the class and set C is consisted of searching keys not belonged to the class. The elements of set B are randomly selected from set A (all existing searching keys). And the elements of set C are selected from the set $A \setminus B$. All selected elements have same probability to be chosen.

We can construct sets of hash values for sets C and B. It is:

$$L = \{1 \le i \le r : B \cap A_i \ne \varnothing\}, \quad M = \{1 \le i \le r : C \cap A_i \ne \varnothing\}$$

$$\lambda = |L|, \ \mu = |M|$$

The condition of collision to come up is following:

$$L \cap M \ne \varnothing \qquad (6)$$

Let's find the probability that intersection of L and M is empty. Let's call "good" any pair of sets B and C if they provide that event. Then probability of collision is a probability of opposite event.

To solve this problem let's solve an auxiliary problem. Let's denote as $t_r(m;l)$ or shorter as t_r the number of m-elements subsets B of set A such that for each $i=1, 2, ..., r$ $B \cap A_i \ne \varnothing$.

Suppose k as minimal integer number greater or equal to $\frac{m}{l}$:

$$k - 1 < \frac{m}{l} \le k \,. \qquad (7)$$

Then $k < 1 + \frac{m}{l} \le m + 1$ or $k \le m$. Let's show that $t_r = 0$ if $r < k$ or $r > m$. And for $k \le r \le m$

$$t_r = (-1)^k \sum_{i=k}^{r} (-1)^i C^m_{(r-i+k)l} C^{i-k}_r \,. \qquad (8)$$

By definition $|A| = rl$. Then with $r<k$ and use (7) $|A| \le (k-1)l < \frac{m}{l} l = m$. It means $t_r = 0$.

If $r > m$, any $B \subset A$ intersecting with every A_i, $1 \le i \le r$ consists not less than r elements and not more m elements. That means $t_r = 0$.

Let's prove (8) for $k \le r \le m$.

If $r=k$ then $t_k = (-1)^k \sum_{i=k}^{k} (-1)^i C^m_{(k-i+k)l} C^{i-k}_k = C^m_{kl}$. This is equal to the number of all possible m-elements subsets of set A (from (7) follows $kl \ge m$). Because of $kl - m < l$ (from (7)) all the subsets intersecting with each A_i, $1 \le i \le k$. Thus we prove (8) for $r=k$.

Let's suppose that (8) is true for any integer $k \le r \le j$. Here j – is integer number form range $k \le j < m$. Let's prove (8) for the case of $r = j+1$. From $rl = (j+1)l$-elements set A we can construct $C^m_{(j+1)l}$ m-elements subsets. For every natural number p let's suppose

$$q_p = \left|\left\{B \subset A : |B| = m, \left|\{1 \le s \le j+1 : B \cap A_s \ne \varnothing\}\right| = p\right\}\right|.$$

It is clear that $q_p = 0$ for $p > j+1$. Besides, because of $t_p = 0$ with $p < k$ (as was proved before) $q_p = 0$ with same p.

Thus

$$C^m_{(j+1)l} = \sum_{p=k}^{j+1} q_p \qquad (9)$$

Every summand in equation (9) may be denoted as

$$q_p = \sum_{\substack{T \subset \{1,2,\ldots,j+1\} \\ |T| = p}} q(T) \qquad (10)$$

Here $q(T) = \left|\left\{B \subset A : |B| = m, B \cap A_i \ne \varnothing \Leftrightarrow i \in T\right\}\right|$.

The sum (10) consists of C^p_{j+1} summands. Every one of them equals t_p. Thus $q_p = C^p_{j+1}$, $p = k,\ldots, j+1$. With (9) and with equation $q_{j+1} = t_{j+1}$ we have:

$$t_{j+1} = C^m_{(j+1)l} - \sum_{p=k}^{j} q_p = C^m_{(j+1)l} - \sum_{p=k}^{j} C^p_{j+1} t_p \,.$$

Using the mathematical induction let's replace r to p within (8):

$$t_{j+1} = C^m_{(j+1)l} - \sum_{p=k}^{j} C^p_{j+1} (-1)^k \sum_{i=k}^{p} (-1)^i C^m_{(p-i+k)l} C^{i-k}_p \qquad (11)$$

Let's replace sum indexes p and i within (11) to $\alpha = j+1-p+i$ and $\beta = j+1-p$. Then we have $p = j+1-\beta$, $i = \alpha - \beta$. Let's solve next system of inequalities:

$$\begin{cases} k \le p \le j \\ k \le i \le p \end{cases}$$

$$\begin{cases} 1 \le \beta \le j+1-k \\ k+\beta \le \alpha \le j+1 \end{cases} \Leftrightarrow \begin{cases} k+1 \le \alpha \le j+1 \\ 1 \le \beta \le \alpha - k \end{cases}.$$

Then (11) can be rewritten as:

$$t_{j+1} = C^m_{(j+1)l} + (-1)^k \sum_{\alpha=k+1}^{j+1} C^m_{(j+1-\alpha+k)l} (-1)^\alpha \times$$

$$\times \sum_{\beta=1}^{\alpha-k} (-1)^{1-\beta} C^{j+1-\beta}_{j+1} C^{\alpha-\beta-k}_{j+1-\beta} \,.$$

The last sum of this expression can be simplified:

$$\sum_{\beta=1}^{\alpha-k} (-1)^{1+\beta} \frac{(j+1)!(j+1-\beta)!}{(j+1-\beta)!\beta!(\alpha-\beta-k)!(j+1-\alpha+k)!} =$$

$$= -\sum_{\beta=1}^{\alpha-k}(-1)^{\beta}\frac{(j+1)!(\alpha-k)!}{(\alpha-k)!(j+1-\alpha+k)!\beta!(\alpha-\beta-k)!} =$$

$$= -\sum_{\beta=1}^{\alpha-k}(-1)^{\beta}C_{j+1}^{\alpha-k}C_{\alpha-k}^{\beta} =$$

$$= C_{j+1}^{\alpha-k}\left(1-\sum_{\beta=0}^{\alpha-k}C_{\alpha-k}^{\beta}(-1)^{\beta}\cdot 1^{\alpha-k-\beta}\right) = C_{j+1}^{\alpha-k}\left(1-(-1+1)^{\alpha-k}\right) =$$

$$= C_{j+1}^{\alpha-k}$$

Thus

$$t_{j+1} = C_{(j+1)l}^{m} + (-1)^{k}\sum_{\alpha=k+1}^{j+1}(-1)^{\alpha}C_{(j+1-\alpha+k)l}^{m}C_{j+1}^{\alpha-k} =$$

$$= (-1)^{k}\sum_{\alpha=k}^{j+1}(-1)^{\alpha}C_{(j+1-\alpha+k)l}^{m}C_{j+1}^{\alpha-k}$$, we have expression (8) with $r = j+1$. Thus expression (8) is true for $k \le r \le m$.

The ranges of possible values of λ and μ are following:

$$k \le \lambda \le \min(r;m),\ s \le \mu \le \min(r-\lambda;n). \qquad (12)$$

Here, s is the least integer that greater or equal of $\frac{n}{l}$. So the $k \le m, s \le n$, then (12) have solution if and only if $\begin{cases} k \le r \\ s \le r-k \end{cases}$, which means

$$k + s \le r \qquad (13)$$

To provide a positive value of collision probability it is necessary and sufficient to satisfy condition (13). Then for each solution (λ,μ) of (12) exists $\frac{r!}{\lambda!\mu!(r-\lambda-\mu)!}$ a pair of sets $L,M \subset \{1,2,...,r\}$ such that $L \cap M = \varnothing$, $\lambda = |L|$, $\mu = |M|$. Each the pair is belonging $t_{\lambda}(m;l)\cdot t_{\mu}(n;l)$ of "good" pair of sets B and C.

So the whole number of such pairs is equal

$$\overline{Q} = \sum_{\lambda=k}^{\min(r;m)}\sum_{\mu}^{\min(r-\lambda;n)}\frac{r!}{\lambda!\mu!(r-\lambda-\mu)!}t_{\lambda}(m;l)t_{\mu}(n;l)$$

Using expression (8) we have

$$\overline{Q} = \sum_{\lambda=k}^{\min(r;m)}\sum_{\mu=s}^{\min(r-\lambda;n)}\frac{r!}{\lambda!\mu!(r-\lambda-\mu)!}\cdot(-1)^{k+s}\times$$

$$\times\sum_{i=k}^{\lambda}(-1)^{i}C_{(\lambda-i+k)l}^{m}C_{\lambda}^{i-k}\sum_{j=s}^{\mu}(-1)^{j}C_{(\mu-j+s)l}^{n}C_{\mu}^{j-s} \qquad (14)$$

Thus we have an expression to find probability of collision for proposed method:

$$P = 1-\frac{\overline{Q}}{W},$$

$$P = 1-\sum_{\lambda=1}^{m}\sum_{\mu=1}^{n}\frac{\prod_{\rho=1}^{\lambda+\mu}(r-(\lambda+\mu)+\rho)}{\prod_{\alpha=1}^{m+n}(rl-(m+n)+\alpha)}\times$$

$$\times\sum_{i=1}^{\lambda}\sum_{j=1}^{\mu}(-1)^{i+j}\frac{\prod_{\beta=1}^{m}((\lambda-i+1)l-m+\beta)}{(i-1)!(\lambda-i+1)!}\frac{\prod_{\gamma=1}^{n}((\mu-j+1)l-n+\gamma)}{(j-1)!(\mu-j+1)!}.$$

Figure 4 shows the results of calculating collision probability for proposed method. The bold lines show values of the probability in dependence on n and m. The diagonal axis s is the sum of n and m. Here we can see that for equal s the worst case in since of collision probability is when n=m.

To compare the probability of collision for convenient and proposed methods we calculate collision probability of lookup-table without keys storing with the same size as in the previous example (256Kbits). The figure 4 shows results of comparison.

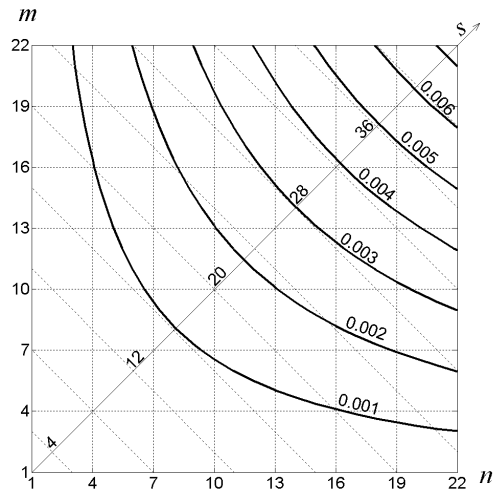

Fig 4. Collision probability for proposed method

On the figure 5 S is a summary number of keys. Line 1, 2 and 3 show dependence of collision probability for convenient method of hashing tables design like in previous example. Line 3, 4 and 5 is for proposed method. Line 4 for the case when n=m. Line 5 for the case when n=8, and line 6 for n=2.

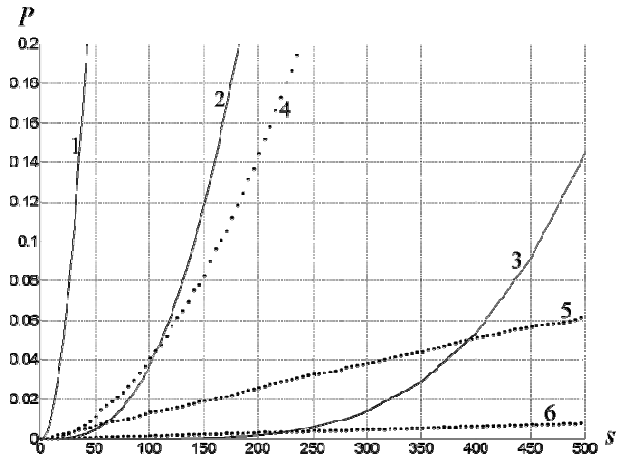

Figure 5. Collision probability comparison

Using of parallel hashing can give dramatic improvement of method efficiency as shown in [8]. The block diagram of frame testing block is shown at figure 6. In the case of using parallel hashing collision came up when we get collision in all of parallel tables.

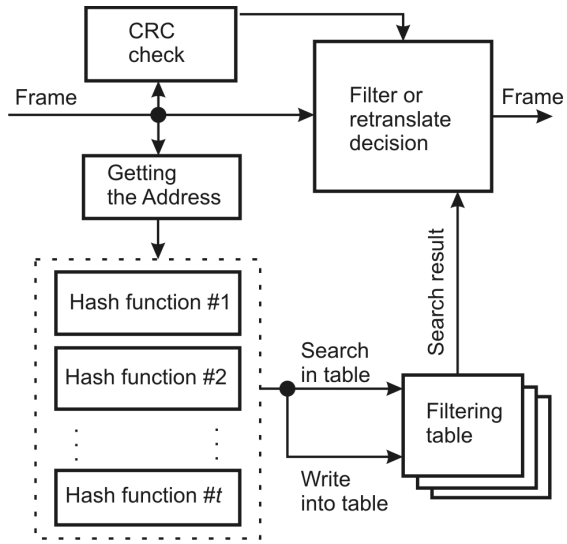

Fig. 6. Structure of Frame testing block

The condition of collision is following:

$$\bigcap_{j=1}^{t} L_j \bigcap M_j \neq \varnothing \qquad (15)$$

According (15) we calculated collision probability and compared it with previous example.

The results are shown on figure 7 were got with the following conditions: 12-bit hash, 8 parallel tables, 4-bit record length in each table, no searching key storing (line 4). Summary required memory size is 128KBit. Line 1 for convenient method (10-bit hash, 64-bit record length, total 256 Kbit). Line 2 – 2 parallel tables, no searching key storing, total 32Kbit. Line 3 – 4 parallel tables, no searching key storing, 64Kbit total.

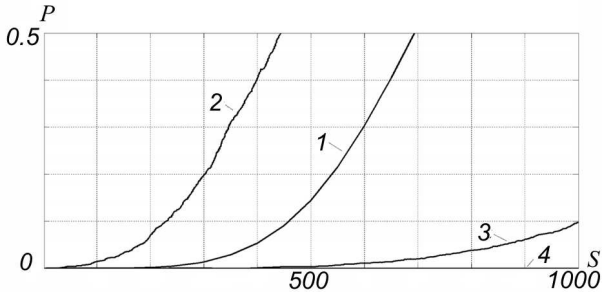

Fig. 7. Parallel hashing collision probability comparison

From figure 7 it is clear that even for 4 parallel tables the collision probability is much lower than for the block method. In addition, the total memory size to allocate the tables for proposed method was per 4 times lower than the reference method.

With the increasing number of parallel hashed tables up to 8, the tables size was 128 Kbit, and the collision probability was lowered to minimal, as for the reference method, the memory size of the table was 8 Mbit meaning a reduction of 64 times in memory usage.

During the studying of relation collision probability of lookup-table table designed by proposed method we found an optimal number of parallel tables for a fixed total memory size. figure 8 shows the dependence of collision probability according to the number of parallel tables. The number of existing keys was S = 1000 and for total memory size of 32 Kbit (line 1), for 64 Kbit (line 2) and for 128 Kbit (line 3). In this figure values near the axis t, is not actually zeros. These are about 10^{-7} and less.

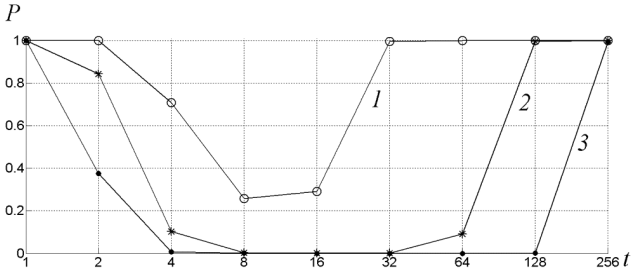

Fig. 8. Optimum of probability

These results suggest an optimal number of parallel tables for every fixed total memory size to allocate parallel hash lookup-tables.

IV. "IDEAL" HASHING

The ideal hashing or perfect hash function allows to avoid collision at all.

We can try to find the perfect hash function by changing the hash calculating way if we detect the collision state. But this approach has a lack. We need to rebuild or refill the table after hash function changing. It may take much time depends on size of table.

To eliminate the lack we propose to combine parallel hashing with a mechanism of perfect hash selection.

One of the parallel hash tables should be to change its hash function. If collision detected, we change way of the hash computing – choose another hash function for this table. The bank of hash functions for choosing for adaptive hashing should not contain the hash functions used in other of parallel hash tables.

Usage of the adaptive hashing gives us geometric distribution of collision probability in depends on number of hash functions in bank of adaptive hash functions v. Collision probability decreasing according following expression

$$P_{adapt.} = P_t^v \qquad (16)$$

here P_t is collision probability for t parallel hash tables before changing the adaptive hash.

Thus, if initial collision probability for proposed method

was 10^{-2}, in case of collision detection, the first iteration of hash function changing will give us 10^{-4}, next iteration - 10^{-6} e.c.t.

As we can see, the required amount of memory is similar to the proposed method of parallel hashing without searching keys storing, but thanks for adaptive mechanism we can decrease collision probability to required value. Besides we insignificantly los the hash table in the process of changing the adaptive hash function. This is because of we have several other parallel tables that keep working during adaptive table is rebuilt.

V. HASH CALCULATION TECHNIQUES

There are many algorithms and techniques of hash calculation. Recent time are using many complex approaches such as CVM, neural networks, etc. [9, 10]. Such complex methods give very good results in classification problems but need huge time for learning.

For the ASIC realization we need simple and very fast technique. The main demands is minimizing of memory usage and time for hash calculating. The simplest way of hash calculation is modulo operation. In this case hash is the reminder of Euclidian division of the transmitted data and some constant. One of realization of this approach is polynomial division or CRC calculating.

The possible number of CRC calculating ways for fixed hash width is limited by this width as 2^{width}. Not all of them are "good" in context of Hemming distance [11, 12, 13].

In this work we used 16 variation of polynomial for from 10 to 16 bits CRC.

Searching of the domain related set of polynomials for CRC calculation is an our further aim.

VI. REALIZATION WITHIN FPGA

For comparison existing device and proposed method we designed a realization for implementing in FPGA (Cyclone II series). We designed three-port switch that is functional analog of ADM6993 by Infinion [14].

For comparison existing device and proposed method we designed a realization for implementing in FPGA (Cyclone II series). We designed three-port switch that is functional analog of ADM6993 by Infinion [14]. Figure 8 and 9 shows block diagram of switch port and frame testing block with using four parallel tables.

Designed device has three ports. One of them was transfer speed limited on the level of 1024 Kbit/s. We loaded switch by short packets and compared power consumption for both devices. Results of experiment are in the Table I.

Both devices – prototype and analog was supplied by DC current 3.3 V. We measured current on the power supply in three modes: for 10% load of limited port, 50% and 100% load.

During whole experiments we controlled loss of packets. We obtained that designed device has at least 2,2 times lover power dissipation then prototype.

TABLE I
POWER CONSUMPTION COMPARISON

Chanal load, %	Power consumption, mW		Ratio
	Prototype	Designed divice	
10	1172	450	2,6
50	1216	521	2,3
90	1228	565	2,2

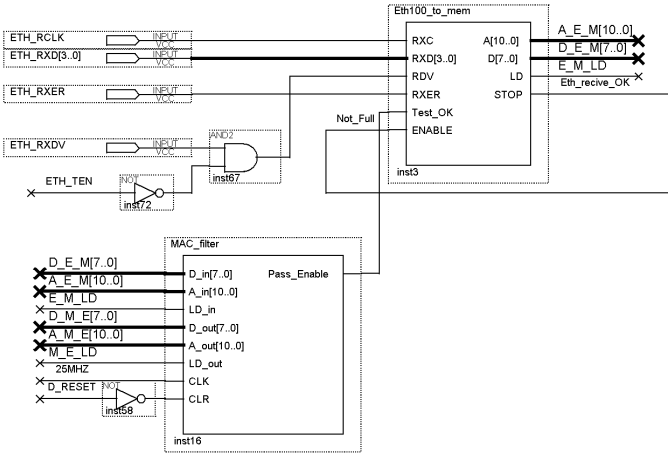

Fig. 8. Switch port block diagram

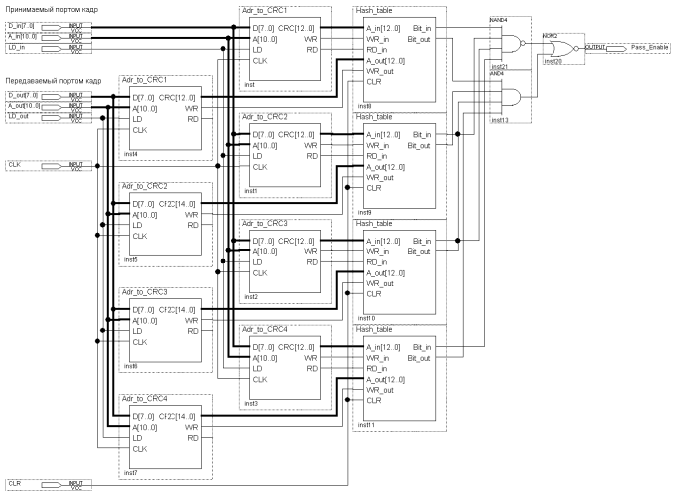

Fig. 9. Frame testing block diagram

VII. CONCLUSIONS

The proposed in the paper approaches to design hash lookup-tables allows dramatic decrease the amount of required memory without losing efficiency of lookup-table performance. Besides, these approaches are ready to be realized as IP-cores or ASIC microchips that can work with much more lower clock frequency and supply voltage. It can reduce the load to the CPU and main memory of mobile device.

Proposed method allowed to reduce the power consumption of network switch at least 2 times in comparison with convenient method. This result was confirmed in stand tests.

VIII. Acknowledgement

The reported study was supported by the Russian Foundation for Basic research (RFBR), research projects №17-57-53192 and №16-37-00386.

References

[1] Huang, Junxian, et al. "A close examination of performance and power characteristics of 4G LTE networks." Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012.

[2] Chandrakasan, Anantha P., and Robert W. Brodersen. Low power digital CMOS design. Springer Science & Business Media, 2012.

[3] Alam, Faisal, et al. "Energy optimization in Android applications through wakelock placement." Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014. IEEE, 2014.

[4] Ma, Yi-Wei, et al. "A Power Saving Mechanism for Multimedia Streaming Services in Cloud Computing." Systems Journal, IEEE 8.1 (2014): 219-224.

[5] Congdon, Paul T., et al. "Simultaneously reducing latency and power consumption in openflow switches." IEEE/ACM Transactions on Networking (TON) 22.3 (2014): 1007-1020.

[6] Knuth, D. "The Art of Computer Programming 1: Fundamental Algorithms 2: Seminumerical Algorithms 3: Sorting and Searching." (1968)

[7] McNeil Jr, Roy, Yilun Wang, and Sheng Wu. "Using CRC-15 as hash function for MAC bridge filter design." U.S. Patent No. 7,620,043. 17 Nov. 2009.

[8] Makov, Sergey V., et al. "A method for ultra fast searching within traffic filtering tables in networking hardware." IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics, 2015.

[9] Wang, Jun, et al. "Learning to hash for indexing big data—a survey." Proceedings of the IEEE 104.1 (2016): 34-57.

[10] Frantc, V. A., et al. "Simultenious binary hash and features learning for image retrieval." SPIE Commercial+ Scientific Sensing and Imaging. International Society for Optics and Photonics, 2016.

[11] Koopman, Philip, and Tridib Chakravarty. "Cyclic redundancy code (CRC) polynomial selection for embedded networks." Dependable Systems and Networks, 2004 International Conference on. IEEE, 2004.

[12] Castagnoli, Guy, Stefan Brauer, and Martin Herrmann. "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits." IEEE Transactions on Communications 41.6 (1993): 883-892.

[13] Koopman, Philip. "32-bit cyclic redundancy codes for internet applications." Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on. IEEE, 2002.

[14] ADM6993/X HDLC to Fast Ethernet converter: data sheet Rev1.11, Nov. 2005 URL: http:// img.chipfind.ru/ pdf/ infineon/ adm6993_x_ds_green_version_1.pdf