Journal of ELECTRICAL ENGINEERING, VOL. 56, NO. 5-6, 2005, 151–155

DISTRIBUTED COMPUTER NETWORK FOR COMPUTATION OF WEIGHT SPECTRA FOR SOME ERROR CONTROL CODES

Martin Rakús — Jozef Oboňa — Peter Farkaš
Ladislav Divinec — Pavol Čižmárik *

Error control codes (ECC) are used today in variety of different applications in the area of communications. Recently some new linear block codes defined over finite fields were constructed. Newly found codes have higher code rate (smaller redundancy) than widely used Reed Solomon codes constructed over the same finite field. Error correcting capabilities of ECC is determined by the code distance d_m. In order to determine the code distance for proposed codes a complete weight spectrum has to be calculated, because analytical solution of weight spectra determination for mention codes does not exist. Since the number of codewords in each of the found codes is very high (up to 16^{235}, if constructed over $GF(16)$) direct calculation of weight spectra of such codes is therefore infeasible. Using the relationships between original code and its dual significantly simplifies the problem, but computational complexity is still too high to be computed on a single PC. Therefore to overcome the computational complexity it was necessary to use some structural knowledge and also to built an appropriate computer network architecture for distributed computing. In this paper the practical approach to calculation of the weight spectra is described.

K e y w o r d s: distributed computing, linear block code, finite field, Reed Solomon codes, code rate, code distance, error control code

1 INTRODUCTION

Ever increasing transfer rate of today's transmission systems requires high performance ECC in order to satisfy the demanding speed and error rate requirements. To protect transmitted or stored information against errors, ECC add redundancy to the transmitted user information. For most applications linear block codes (LBC) are used because they have known coding algorithms with acceptable complexity. LBC is usually defined as a k-dimensional subspace of n-dimensional vector space over a finite field. Error correcting and detecting capability of ECC is determined by the so-called code distance d_m. For a high speed transmissions besides error correcting capabilities of chosen ECC another parameter called code rate denoted as R is important as well. R equals to the ratio of the number of information symbols to the codeword length in LBC. It represents the redundancy added by the ECC to user information. The ideal ECC should have the number of correctable errors denoted as t as high as possible and R approaching 1. Those two condition obviously contradict so in a real application some compromise has to be chosen. Our goal was to find codes having the highest possible value of R (the smallest redundancy) while maintaining error correcting capabilities comparable to popular Reed Solomon codes (if constructed over the same finite field $GF(q)$). In order to determine t of newly found codes an accurate calculation of d_m was necessary. Determination of d_m for high speed codes with a high code rate is computationally very demanding. Carrying out precise calculation of d_m for such codes executable in "reasonable" time span was the main motivation for the development of computer network architecture for distributed computing presented in this paper.

2 BASIC THEORY OF LINEAR BLOCK CODES

Let $C = [n, k, d_m]$ be a linear block code defined over finite field $GF(q)$, where $q = 2^r$. r is the number of bits contained in one codeword symbol. n denotes the number of codeword symbols, k denotes the number of information symbols and d_m denotes the code distance. Important terms in coding theory are *Hamming distance* and *Hamming weight*. Hamming distance of two vectors $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ and $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$ is the number of symbols in which these two vectors differ. It is denoted as $d(\boldsymbol{v}, \boldsymbol{u})$. By minimal or *code distance* is denoted the minimal Hamming distance between any two codewords of a given code. The code distance is denoted as d_m. A linear block code (LBC) is capable of correcting t errors if $d_m \geq 2t+1$. The hamming weight of vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ is the number of its non-zero elements. It is denoted as $w(\boldsymbol{v})$. The linear block code is completely defined by its generator matrix $\mathbf{G}$ with dimensions $k \times n$. LBC has a generator matrix in the form of $\mathbf{G} = [\mathbf{I}|\mathbf{A}]$ where $\mathbf{I}$ is the identity matrix with dimensions $k \times k$ and $\mathbf{A}$ is parity matrix with dimensions $k \times (n-k)$. LBC generated by such $\mathbf{G}$ matrix is called *systematic code*. In [1] it was shown that any non-systematic LBC can be

Slovak University of Technology, Faculty of Electrical Engineering and Information Technology, Department of Telecommunications, Ilkovičova 3, 812 19 Bratislava, Slovakia, E-mails: rakus@ktl.elf.stuba.sk, p.farkas@ieee.org, ld99121@decef.elf.stuba.sk

ISSN 1335-3632

transformed to a systematic form by appropriate matrix manipulation on $\mathbf{G}$. In order to systematically encode an information vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ it is necessary to perform to following multiplication $\boldsymbol{c} = \boldsymbol{u} \cdot \mathbf{G}$, where $\boldsymbol{c}$ is a code word vector:

$$(u_1, u_2, \ldots, u_k) \cdot \begin{bmatrix} 1\ 0\ 0\ \ldots\ 0 & a_{11} & \ldots & a_{1n-k} \\ 0\ 1\ 0\ \ldots\ 0 & a_{21} & \ldots & a_{2n-k} \\ & & \vdots & \\ 0\ 0\ 0\ \ldots\ 1 & a_{k1} & \ldots & a_{kn-k} \end{bmatrix} = (u_1, u_2, \ldots, u_k, v_{k+1}, \ldots, v_n)$$

where vector $\boldsymbol{v} = (v_{k+1}, \ldots, v_n) = \boldsymbol{u}\mathbf{A}$. In the case that for encoding instead of $\mathbf{G}$ matrix the control matrix $\mathbf{H}$ is used, a so called *dual code* is generated. Control matrix $\mathbf{H}$ has the following property: $\mathbf{H} = \left[\mathbf{A}^\top | \mathbf{I}\right]$, where $\mathbf{A}^\top$ is transposed parity matrix $\mathbf{A}$ and $\mathbf{I}$ is the identity matrix. $\mathbf{H}$ has dimensions $(n-k) \times n$. If the Hamming distance of two codewords $\boldsymbol{v}$ and $\boldsymbol{v}'$ equals d, then $w(\boldsymbol{v} - \boldsymbol{v}') = d$. It means that the code distance of LBC d_m equals to the minimal weight of its non-zero codeword. It implies that the exact knowledge of weight distribution of all codewords by their weights called *weight spectrum* allows to determine exactly d_m and thus to know t. Several methods exist for the calculation of weight spectrum:

- analytical methods. For certain types of codes exact formulas for the weight spectra calculation based on $[n, k]$ parameters and type of used $GF(q)$ were derived. A typical example is a group of MDS codes to which belong the well known Reed-Solomon codes.
- direct calculation of weights of all codewords.
- statistical methods. This approximate method applies only for certain types of codes.

3 DOUBLE ERROR CORRECTING CODES WITH HIGH CODE RATE

In this section a new family of codes and one example from that family a [235,225,5] code constructed over $GF(16)$ will be presented. These codes could be constructed over any finite fields $GF(q)$, where $q = 2^r$, $r \geq 3$. They have a higher code rate than Reed Solomon codes if constructed over the same finite fields. In contrast to conditionally double error correcting codes defined in [3], the new codes have a code distances, which allow to make exact conclusions about their error control capabilities. A family of codes that use polynomial arithmetic and process r bit symbols was introduced in [2]. Symbols $Q_1, Q_2, \cdots, Q_{q-2}, P_0, P_1, \cdots, P_{z-1}$ are binary strings of length r and each Q_i and P_i can be regarded as a member of $GF(q)$ where $q = 2^r$. α is a primitive element of the finite field $GF(q)$. $Q_{q-2}, Q_{q-3}, \cdots, Q_1$ is a message consisting of $q-2$ symbols from $GF(2^r)$ and it is encoded as a string of n symbols, where n is a codeword length, by adding the check symbols $P_0, P_1, \cdots, P_{z-1}$. Check symbols are calculated by equation (3). The new family of codes is defined by $\mathbf{H}$ matrix (1)

$$\mathbf{H} = \left[\begin{array}{cccc|c} \multicolumn{4}{c|}{\mathbf{A}} & \cdots \\ \alpha^{q-2} & \cdots & \alpha^{q-2} & \alpha^{q-2} & \cdots \\ \alpha^{2(q-2)} & \cdots & \alpha^{2(q-2)} & \alpha^{2(q-2)} & \cdots \\ \alpha^{3(q-2)} & \cdots & \alpha^{3(q-2)} & \alpha^{3(q-2)} & \cdots \\ \alpha^{0} & \cdots & \alpha^{3} & \alpha^{2} & \cdots \\ \alpha^{q-3} & \cdots & \alpha^{1} & \alpha^{0} & \cdots \\ \alpha^{0} & \cdots & \alpha^{q-4} & \alpha^{q-3} & \cdots \end{array}\right.$$

$$\left.\begin{array}{c|ccc|ccc|c} & \multicolumn{3}{c|}{\mathbf{A}} & \multicolumn{3}{c|}{\mathbf{A}} & \\ \alpha^{q-2} & \cdots & \alpha^{1} & \alpha^{0} & \alpha^{q-2} & \cdots & \alpha^{2} & \alpha^{1} & \alpha^{0} \\ \alpha^{q-1} & \cdots & \alpha^{2} & \alpha^{1} & \alpha^{q-1} & \cdots & \alpha^{3} & \alpha^{2} & \alpha^{1} \\ \alpha^{0} & \cdots & \alpha^{3} & \alpha^{2} & \alpha^{1} & \cdots & \alpha^{q-3} & \alpha^{q-2} & \alpha^{0} \\ \alpha^{q-1} & \cdots & \alpha^{q-3} & \alpha^{q-2} & \alpha^{q-1} & \cdots & \alpha^{q-4} & \alpha^{q-3} & \alpha^{q-2} \end{array}\right] \mathbf{I}$$

$$= [\mathbf{B}\,\mathbf{I}] \quad (1)$$

where submatrix $\mathbf{A}$ has form:

$$\mathbf{A} = \begin{bmatrix} 1 & \cdots & 1 & 1 & 1 \\ \alpha^{q-2} & \cdots & \alpha^{2} & \alpha^{1} & \alpha^{0} \\ \alpha^{2(q-2)} & \cdots & \alpha^{4} & \alpha^{2} & \alpha^{0} \\ \alpha^{3(q-2)} & \cdots & \alpha^{6} & \alpha^{3} & \alpha^{0} \end{bmatrix}. \quad (2)$$

$$\begin{aligned} P_9 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} Q_i \\ P_8 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{i} Q_i \\ P_7 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{2i} Q_i \\ P_6 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{3i} Q_i \\ P_5 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{(q-m)} Q_i \\ P_4 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{2(q-m)} Q_i \\ P_3 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{3(q-m)} Q_i \\ P_2 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{(i+m)-1} Q_i \\ P_1 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{(i-m)+1} Q_i \\ P_0 &= \textstyle\sum_{m=2}^{q} \sum_{i=u}^{v} \alpha^{1-(i+m)} Q_i \end{aligned} \quad (3)$$

In order to determine the error correcting capabilities of the proposed new family of codes it was necessary to calculate its weight spectra to obtain d_m. The proposed family of codes does not belong to any category of codes for which an exact analytical formula for calculation of weight spectra was derived. So the only solution is to calculate the complete weight spectra by generating all codewords. The number of codewords to be generated is in general q^k. In our case $q^k = 16^{225}$. But the existence of our Universe is estimated to be "only" about 4×10^{17} seconds $\ll 16^{225}$. The direct calculation of weight spectra for such codes is therefore infeasible. The solution is to use the relationship between weight spectra of the normal code and its dual. These relationship is determined by Krawtchouk polynomials or by McWilliams identities [4]. The relationship between weights of the normal code and its dual defined by Krawtchouk polynomials is defined by the following equation:

$$A_k = \frac{1}{|\mathcal{C}^\perp|} \sum_{i=0}^{n_d} A_i^\perp P_k(i) \quad (4)$$

where:

A_k is the number of codewords of weight k of normal code,

$A_i^{\perp}$ is the number of codewords of weight i of dual code,

n_d is the codeword length of a dual code,

$|\mathcal{C}^{\perp}|$ is the number of codewords of a dual code

$P_k(i)$ is Krawtchouk polynomial of dimension k of variable x.

For non binary codes Krawtchouk polynomials of degree k for any positive integer n, where x is a variable are defined as follow:

$$P_k(x,n) = P_k(x) = \sum_{j=0}^{k} (-1)^j (q-1)^{k-j} \binom{x}{j} \binom{n-x}{k-j}, \quad k = 0, 1, \ldots, n. \quad (5)$$

In order to calculate the weight spectra of a normal code it is necessary to calculate the weight spectra of its dual code. Dual code has in general q^{n-k} codewords. In our case $q^{n-k} = 16^{10}$ codewords. This number is still too high to be calculated on a single PC, therefore a system for distributed computing specialized for the calculation of weight spectra was developed.

An example of the relevant part of the beginning of the weight spectra of the proposed $[235, 225, 5]_{16}$ code is given bellow:

$$\begin{aligned} A_0 &= 1, & A_7 &= 1792117575 \\ A_5 &= 746550, & A_8 &= 497616264900 \\ A_6 &= 25247250, & A_9 &= 181009559619600 \\ & & & \vdots \end{aligned}$$

Minimal nonzero weight is 5 so the proposed codes are capable of double error correction.

4 SYSTEM FOR DISTRIBUTED COMPUTING

The backbone of the developed system for distributed computing are 6 PCs with CPU type AMD Athlon XP connected to the local LAN through 100Mb Ethernet switch. The system is working on the server-client basis. During the design the cost of the complete system had to be taken into account, therefore we chose Linux for OS. Thanks to this the client consists only from motherboard with CPU and RAM, NIC and case with the power supply. Bootsequence for clients is stored in BootRAM in their NICs. After boot clients download a simple version OS from the network. On the server, besides OS Linux a software LTSP (Linux Terminal Server Project) is installed. This software provides uploading of OS on clients. This way we saved on hardware (HDD, graphic cards, keyboards ...) and software. The realized system for distributed computing is very simply scalable just by adding new clients to the system. All necessary adjustments are performed automatically by software.

A complex task in the system for distributed computing is to efficiently distribute a given task to smaller tasks which are independent of results from other tasks. If this requirement can not be fulfilled then it can happen that one client will be waiting for another to finish its task. The other limiting factor is the communication between the server and client. The amount of this communication has to be kept to minimum in order not to slow down the computation. The ratio between the useful computing time and the time spent for communication has to be $\gg 1$. On the other hand if tasks are too small, then it can happen that sending results to server and downloading a new task takes comparable amount of time as computing of that task itself. Splitting the big tasks to smaller tasks effectively is therefore a complex problem. In our case this was fortunately trivial. We need to calculate the weight spectra of 16^{10} vectors, therefore clients need only the range of vector for which the weight spectrum has to be calculated. Splitting the task into smaller tasks is performed by the server, where a software generating an input data for clients is running. Since clients in our system have the same computing power, the range of information vectors was set the same for all clients. After generating an input file of data for the client this file is sent to the client by means of sockets. Client stores the input information into a file which servers as an input parameter for a software performing the actual calculation of weight spectra. When the client finishes its task it activates a connection with the server and sends server the result. The server keeps track about already finished tasks in a special created table. When all partial tasks are done, the server combines the complete weight spectrum out of the received output files sent from clients. During calculation of the weight spectra of codes with long codewords it can happen that variables type *long* in which are stored the number of already encoded vectors as well as particular weights can overflow. To overcome this problem we are storing those two variables as type *string* and then perform addition. This way we can analyze also codes with really long codewords at the price of a slight slowdown.

The process of encoding was described in section 2. This comes down to left multiplication of information vector with generator matrix $\mathbf{G}$. Elements of matrices and vectors are symbols of finite fields with the base 2 denoted as $GF(2^r)$. The advantage of using the finite field is their closure property, it means that operations of multiplication and addition always give an element from a given field. Therefore pre-computed tables of addition and multiplication can be used to speed up the calculation by means of reading out results from the table using a coordinate system where the intersection gives the results

Table 1.

dual code parameters $[n,k]$	# of operations: original algorithm	# of operations: modified algorithm	ratio
(55,6) over GF(8)	154 140 672	33 029 430	4.7
(57,8) over GF(8)	13 153 337 344	2 113 928 306	6.2
(228,3) over GF(16)	5 529 600	2 087 100	2.6
(230,5) over GF(16)	2 359 296 000	534 771 000	4.4

of the operation above particular elements. Eg. table for addition over $GF(8)$:

$$\begin{array}{cccccccc} -1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & -1 & 3 & 6 & 1 & 5 & 4 & 2 \\ 1 & 3 & -1 & 4 & 0 & 2 & 6 & 5 \\ 2 & 6 & 4 & -1 & 5 & 1 & 3 & 0 \\ 3 & 1 & 0 & 5 & -1 & 6 & 2 & 4 \\ 4 & 5 & 2 & 1 & 6 & -1 & 0 & 3 \\ 5 & 4 & 6 & 3 & 2 & 0 & -1 & 1 \\ 6 & 2 & 5 & 0 & 4 & 3 & 1 & -1 \end{array}$$

where elements of the table represent the powers of primitive element α, (-1 represents 0). Eg. $\alpha^2 \oplus \alpha^3 = \alpha^5$.

The algorithm for determining the code distance d_m has the following steps:

- generation of information vector,
- encoding of information vector by its left multiplication with generator matrix,
- determining the weight of codeword,
- incrementing the particular weight in weight spectrum.

The main problem of this algorithm is its exponentially growing complexity, since the number of codewords is growing exponentially with the number of information symbols q^k. Therefore we looked for the way how to make this algorithm more efficient. The number of additions and multiplications necessary to encode one information vector is $2kn$. If we take into account that an identity matrix of $\mathbf{G}$ only copies the information vector, then the number of operations can be reduced to $2k(n-k)$. Out of this number one half of operations belong to addition and the other half to multiplication. Since both operations can be performed as a simple access to 2d field, they require the same amount of time for execution. Since it is necessary to encode all information vectors, we tried to reduce the number of operations necessary to encode each single information vector. The reduction of operations will be illustrated on the following example. Let $k = 6$ and let $GF(q) = GF(8)$. The first information vector is $\alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{-1}$. The first 6 elements of code word is the information vector itself. In the next the information vector is multiplied only with the parity matrix $\mathbf{P}$ of $\mathbf{G}$:

$$\begin{aligned} &\alpha^{-1}p_{11} + \alpha^{-1}p_{21} \\ &\quad +\alpha^{-1}p_{31} + \alpha^{-1}p_{41} + \alpha^{-1}p_{51} + \alpha^{-1}p_{61} = \boldsymbol{v}_{k+1} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ &\alpha^{-1}p_{1(n-k)} + \alpha^{-1}p_{2(n-k)} + \alpha^{-1}p_{3(n-k)} \\ &\quad +\alpha^{-1}p_{4(n-k)} + \alpha^{-1}p_{5(n-k)} + \alpha^{-1}p_{6(n-k)} = \boldsymbol{v}_n \end{aligned}$$

The next information vector is $\alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{0}$. At the multiplication with the parity matrix $\mathbf{P}$ can be seen that the first 5 elements remained unchanged that means their multiplication with $\mathbf{P}$ is identical as is in the step before. The change will take place after addition of $\alpha^0 p_{61}$. Therefore we store temporary sums:

$$\begin{aligned} &\alpha^{-1}p_{11} \\ &\quad +\alpha^{-1}p_{21} + \alpha^{-1}p_{31} + \alpha^{-1}p_{41} + \alpha^{-1}p_{51} = temp_1 \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ &\alpha^{-1}p_{1(n-k)} + \alpha^{-1}p_{2(n-k)} \\ &\quad +\alpha^{-1}p_{3(n-k)} + \alpha^{-1}p_{4(n-k)} + \alpha^{-1}p_{5(n-k)} = temp_{n-k} \end{aligned}$$

from the calculation of the previous codeword. In this way we save 4 additions and 5 multiplications per each symbol, because in order to calculate $\boldsymbol{v}_{k+1}$ we only need to add $\alpha^0 p_{61}$ to $temp_1$ to get the result. We used a similarity of successive information vectors. This way we can easily encode information vectors:

$$\begin{array}{cccccc} \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{0} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{1} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{2} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{3} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{4} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{5} \\ \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{-1} & \alpha^{6} \end{array}$$

The next vector is $\alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{-1}, \alpha^{0}, \alpha^{-1}$. Since the 5th position has been changed, we can not use $temp_1, \ldots, temp_{n-k}$ from the previous encoding. The next step is a calculation of $temp$ for the new information vector. This operation can be simplified as well if we add to the previous $temp_1$ again $\alpha^{-1}p_{51}$, we get: $temp_1 = \alpha^{-1}p_{11} + \alpha^{-1}p_{21} + \alpha^{-1}p_{31} + \alpha^{-1}p_{41}$. If we add to this $temp_1$ $\alpha^0 p_{51}$, we get a new $temp_1$, what enables us to encode the next 8 vectors. This is possible because in finite fields of base 2 the operation of addition is identical to subtraction. Table 1 proves the advantages of the proposed algorithm. Flowcharts for server and client applications are shown in Fig. 1 and Fig. 2 respectively.

5 CONCLUSION

Design and realization of the system for distributed computing enabled us to calculate exact weight spectra of codes with big codeword length for which this calculation on a single PC would be out of reasonable time bounds. The next step in the development will be the modification

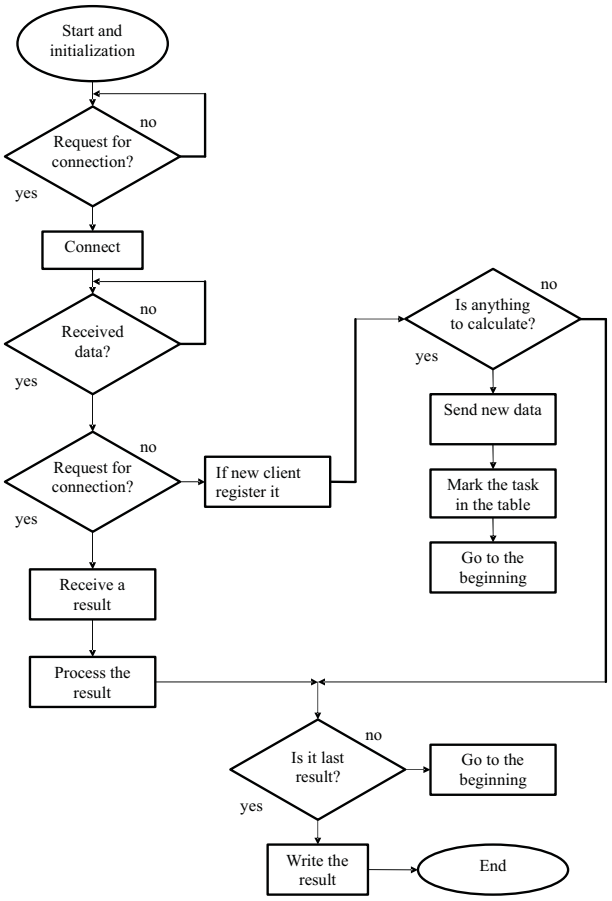

Fig. 1. Flowchart of server application

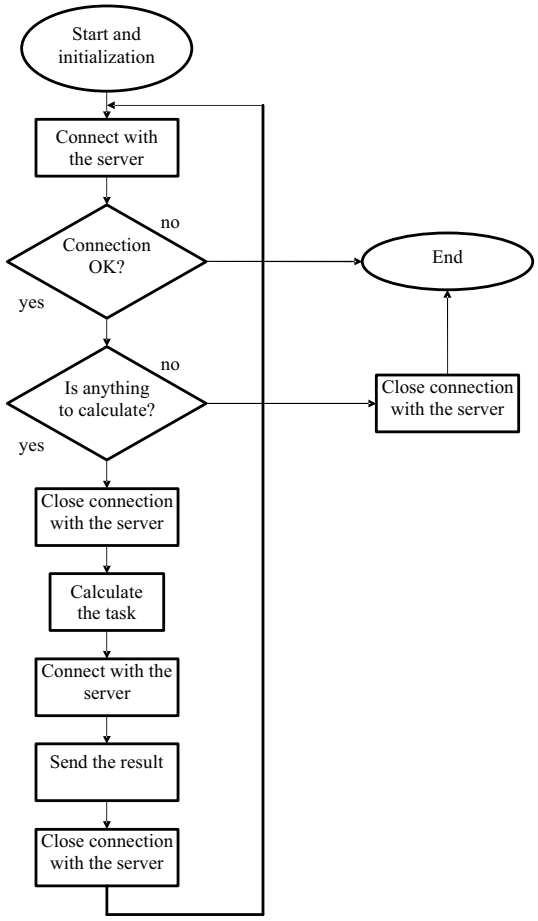

Fig. 2. Flowchart of client application

of the existing system for use in a heterogenous network of PCs connected together through distant LANs.

References

[1] BLAHUT, R. E.: Theory and Practice of Error Control Codes, Addison–Wesley Publishing Company, 1984.

[2] McAULEY, A. J.: Weighted Sum Codes for Error Detection and their Comparison with Existing Codes, IEEE Trans. on Networking **2** No. 1 (1994), 16–22.

[3] FARKAŠ, P.—BAYLIS, J.: Modified Generalized Weighted Sum Codes for Error Control, Coding, Communications and Broadcasting, Research studies Press LTD., Baldock, Hertfordshire, England, 2000, pp. 63–72.

[4] McWILLIAMS, F. J.—SLOANE, N. J. A.: The Theory of Error-Correcting Codes, 10.imp. North–Holland Mathematical Library, vol. 16. 1998.

Received 21 October 2004

Martin Rakús (Ing, PhD) graduated in radio and electronics from Slovak University of Technology in 2001. Since 1995 he is with the Department of Telecommunications. Currently he is working in the same dept. as an assistant professor. In 2004 he received PhD degree from Slovak University of Technology. His primary research interests are error control coding and communication systems.

Jozef Oboňa (Ing) graduated in informatics from Slovak University of Technology in 2002. Since 2001 he is with the Department of Telecommunications. Currently he is working in the same department as an assistant professor. Since 2002 he has been a PhD student at the Dept. of Telecommunications. His primary research interests are space-time codes, channel modelling and error control coding.

Peter Farkaš (Prof, Ing, DrSc), (MIEEE, MIEICE, URSI) received the Ing (MSc) degree from Slovak University of Technology (STU) in 1980, PhD degree from St Petersburg State Polytechnic University (former Leningrad Polytechnic Institute) in 1987 and DrSc degree from STU in 1997. He is with Department of Telecommunications at STU, since 1999 as a Professor. His research interests include Error Control Coding, Communications Theory, Mobile Communications and MC CDMA.

Ladislav Divinec(Ing) graduated in telecomunications from Slovak University of Technology in 2005. His thesis was about exploitation of parallel processing of computations in searching for new error correcting codes. He is currently with GC System Inc.

Pavol Čižmárik (Ing) graduated in telecomunications from Slovak University of Technology in 2005. His thesis was about exploitation of parallel processing of computations for an analysis of error control codes with large number of codewords. Since 2002 he is working as technical administrator at Faculty of Physical Education and Sport, Comenius University.