Impact of priority levels on the efficiency of priority switching

KAROL MOLNÁR, PAVEL RAJMIC
Department of Telecommunications, Faculty of Electrical Engineering and Communication
Brno University of Technology
Purkyňova 118, Brno
CZECH REPUBLIC

Abstract: - The paper deals with a neural network controlled switch fabric with frame prioritization support. The impact of priority levels on the functionality and efficiency of this switch fabric was deeply investigated. The results of the studies related to the impact of the amount of priority levels are published in this paper.

Key-Words: - Hopfield neural network, switch fabric, prioritization, priority levels

1 Introduction

Our first results in designing a neural network controlled switch fabric with priority support were published in [1]. The model presented there was further analyzed and optimized. The analysis, focused on the impact of the amount of priority levels on the system, just has been finished and evaluated. This paper summarizes the results.

The paper is organized in the following way: In the next chapter the priority switching problem is introduced form the mathematical point of view. The third chapter introduces the switch fabric model used during our analysis. The fourth chapter describes the requirements on priority level analysis, and the fifth chapter summarizes the results.

2 Priority switching

In the case of classical data switches all the frames are of the same priority. This means that a classical switch is not able to process selected frames prior to processing others. The processing order of the frames is derived from their receiving order. This property has a very undesirable effect on real-time network applications. In the case of such an application, correct timing is essential and blocking the frames in the switch fabric can cause an incorrect function of the application.

Modern multimedia applications require the network to support Quality of Service (QoS). The physical realization of QoS support in a switch can be realized by assigning different priority levels to the frames received. The assignment procedure depends on the QoS technology implemented and is beyond the of scope of this work. Based on the assigned priority the switch can decide which frames must be processed first. Since the switch has usually several ports, the decision process is quite complex. During our work a switch with a crossbar switch fabric and input buffers was modeled.

2.1 Mathematical model of the switch

The input buffer of each port of the switch can be described by a vector. Each element of the vector corresponds to one output port in the switch. This means that the number of elements in the vector is equal to the number of ports. The first element corresponds to the first output port, the second element to the second one, etc. The value of the elements is equal to the priority of the frame that must be forwarded to this port. If there is no frame for the given output port, the element will be equal to the lowest priority level.

For example, if 0 corresponds to the highest and 255 to the lowest priority, vector (0, 5, 255, 255) means that in a four-port switch there is buffer that contains a frame with priority 0, which must be forwarded to the first output port, and another frame with priority 5, which must be forwarded to the second port. There are no frames for the third and fourth output port. In this example there are 256 priority levels. The impact of the amount of priority levels on the efficiency is the main interest of this work.

Each port of the switch can be described in this way. Collecting these vectors we get a matrix expressing the recent state of all buffers. This matrix will be used for our optimization process and it will be called *optimization matrix* and marked **C**.

2.1 The optimization process

The aim of the optimization process is to find the optimal combination of frames that can be sent out through the output ports. This means that we must derive from the optimization matrix a *configuration matrix* describing the optimal configuration (on and off states) of the switches in the switch fabric. The configuration matrix can be thought of as a filter matrix containing elements with value 1 in the places corresponding to the selected elements in the optimization matrix and value 0 for all the other elements. It is important to realize that at one time only one frame can be forwarded from one input to one output. Thus, the configuration matrix must contain just one element with value 1 (on-state of the switch) in each row and each column. These limitations express the confinement criteria of the priority switching problem.

There are several combinations fulfilling these confinement criteria. From this set of valid solutions we must select the best one. If 0 is assigned to the highest priority and larger values correspond to lower priorities, it is considered to be the best solution when the sum of the selected priority values is minimal.

Works [2] and [3] contain useful information on how to express confinement criteria so that they may be suitable for processing by the Hopfield neural network.

The Hopfield neural network is based on solving an iteration process. In our case the result of this iteration process is a configuration matrix, containing only values 0 and 1 and fulfilling the previous confinement criteria. This configuration matrix is the final state of the neural network *state matrix* **V** updated in each iteration cycle. Before the first iteration step this state matrix is generated randomly.

The state matrix **V** can be transformed into a *state vector* **v** using operation *vec*. Operation *vec* takes the columns of the matrix in the argument and arranges them one by one in vertical direction into a vector. The relation between output matrix **V** and the vector form of the output can be expressed by (1):

$$\mathbf{v} = vec\,(\mathbf{V}) = [v_1, v_2, \ldots, v_n]^{\mathrm{T}} \tag{1}$$

$$v_i \geq 0 \ \text{ for all } \ i \in <1; n>$$

Using the state vector **v** the confinement criteria can be expressed in matrix form by (2):

$$\mathbf{A}.\mathbf{v} = \mathbf{b} \tag{2}$$

In the case of the priority switching problem the dimension of matrix **A** will be $2n \ x \ n^2$, where n is the number of the ports. The structure of the first n rows is as follows: the first row will start with a block of n ones continued with n-1 blocks of n zeros. The second row will start with a block of n zeros, continued with a block of n ones and n-2 blocks of n zeros. The third row will start with 2 blocks of n zeros, a block of n ones and n-3 blocks of n zeros, etc. Beginning with the $(n+1)^{\mathrm{th}}$ row the structure will be different. In the $(n+1)^{\mathrm{th}}$ row there will be ones at the 0^{th}, n^{th}, $2n^{\mathrm{th}}$, etc. positions. In the $(n+2)^{\mathrm{th}}$ row there will be ones at the 1^{st}, $(n+1)^{\mathrm{th}}$, $(2n+1)^{\mathrm{th}}$, etc. positions. All the other elements will be zero. This structure is shown in (3).

$$\mathbf{A} = \begin{pmatrix}
1 & 1 & .. & 1 & 0 & \ldots & & & & & & 0 \\
0 & \ldots & & 0 & 1 & 1 & \ldots & 1 & 0 & \ldots & & 0 \\
. & & & & & & & & & & & . \\
. & & & & & & & & & & & . \\
. & & & & & & & & & & & . \\
0 & \ldots & & & & & \ldots & 0 & 1 & 1 & \ldots & 1 \\
1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & 0 \\
. & . & & & & . & & & & . & & . \\
. & & . & & & & . & & & & . & . \\
. & & & . & & & & .. & & & & . \\
0 & \ldots & & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1
\end{pmatrix} \tag{3}$$

All elements in vector **b** will be ones.

$$\mathbf{b} = (1, 1, \ldots, 1)^{\mathrm{T}} \tag{4}$$

If equation (2) is valid, the first n rows of matrix **A** will guarantee that only one active element will be in each row of the output matrix. The following n rows will guarantee that only one active element will be in each column of the output matrix, and their combination into matrix **A** will guarantee that the total number of active elements will be n. In many cases the rows of matrix **A** will be linearly dependent. Since the determinant of matrix **A** will be calculated later, the number of rows of matrix **A** (and vector **b** correspondingly) must be reduced to make them linearly independent.

The object function of the optimization problem is used to evaluate the suitable solutions. Depending on the optimization problem the object function is either minimized or maximized. In our case the object function equals the sum of the priorities of the frames selected for transfer. During the solution of the priority switching problem we seek an object function with the lowest value. The object function $f(\mathbf{v})$ can be expressed as

$$f(\mathbf{v}) = \mathbf{c}^{\mathrm{T}}\mathbf{v} \tag{5}$$

where vector **c** contains weights assigned to the corresponding frames. Vector **c** is derived from the optimization matrix **C** using operation *vec*. If the frame is selected, the corresponding weight value is added to the total sum.

From the knowledge of matrix **A** (3) and vector **b** (4) the transformation function (6) can be constructed

$$\mathbf{v} \leftarrow T_{zs}.\mathbf{v} + \mathbf{s}\,. \tag{6}$$

Transformation matrix $\mathbf{T}_{\mathrm{zs}}$ is defined by (7) and vector **s** is defined by (8).

$$\mathbf{T}_{zs} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} \quad (7)$$

$$\mathbf{s} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \quad (8)$$

Transformation matrix (6) ensures the convergence of the iteration process to a solution fulfilling the confinement criteria. Of course, fulfilling the confinement criteria is not enough. The solution also must minimize object function (5). This is achieved when the state vector **v** is updated during the iteration process by $d\mathbf{v}$, where

$$d\mathbf{v}/dt = \mathbf{T}_{op}.\mathbf{v} + \mathbf{i}_{op} \quad (9)$$

and $\mathbf{T}_{op}$ and $\mathbf{i}_{op}$ are expressed by (10) and (11)

$$\mathbf{T}_{op} = \gamma\,(\mathbf{T}_{zs} - \mathbf{I}) \quad (10)$$

$$\mathbf{i}_{op} = \gamma\mathbf{s} - \mathbf{c} \quad (11)$$

More details about these transformations can be found in [2] and [3].

3 Neural network controlled switch fabric

A neural network controlled switch architecture is shown in Fig 1. Such an architecture but with another type of neural network was presented in [4].

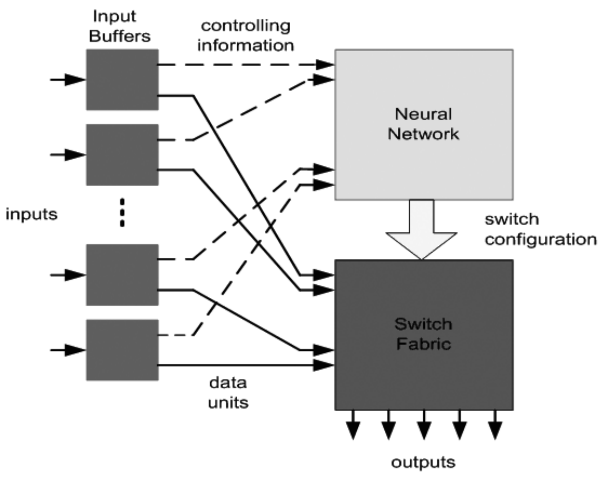

Fig. 1 Neural network controlled frame switch

From the previous chapter it can be seen that the neural network realizes two separate operations. The first operation is related to the confinement criteria and the second is related to the object function. The neural network can be modelled by a block diagram shown in Fig. 2.

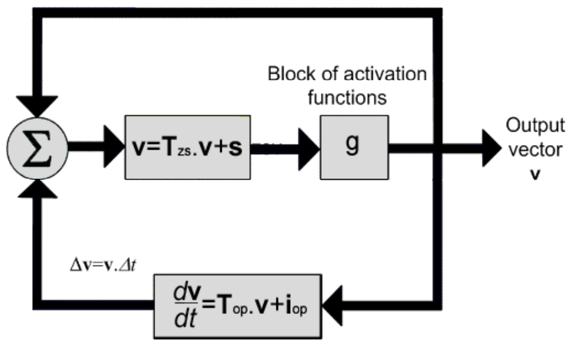

Fig. 2 Block diagram of the neural network

The block diagram of the neural network consists of two loops. The upper loop is responsible for the confinement criteria and the lower loop is responsible for minimizing object function (5). The operations related to the confinement criteria can be separated into two parts. The first part, defined by (6), ensures that the sum of all elements of the state matrix **V** in each row and each column will be equal to 1. The second part, the neurons activation function (7)

$$g(u_{xi}) = \begin{cases} 0 & \text{for} \quad u_{xi} < 0 \\ u_{xi} & \text{for} \quad 0 \le u_{xi} \le 1 \\ 1 & \text{for} \quad u_{xi} > 1 \end{cases} \quad (7)$$

ensures that the output values will remain in the range <0; 1>.

4 Analysis of priority levels

The aim of the analysis was to specify the impact of the amount of priority levels on the efficiency of the iteration process. There are two contradictory requirements on the amount of priority levels. First, the larger the amount of priority values the finer the division of the traffic processed by the switch. From another point of view, increasing the number of priority levels increases the hardware requirements. It also substantially increases the number of required iteration steps. This leads to an increase in operation time, which is very critical in the case of fast frame switching.

As it came out from the results of the analysis, an extremely small amount of priority values messes up the algorithm and leads to invalid solutions. The amount of priority levels was determined by the number of bits used to express the elements of optimization matrix **C**.

The Matlab environment was used to create the simulation model used for the analysis. The results generated by the Matlab model were evaluated from two points of view. First, the number of iteration steps needed to converge to a stable state value was evaluated.

Second, the successfulness of the iteration process was tested. The iteration process was considered successful if the generated result fulfilled the criteria specified earlier.

5 Impact of the amount of priority levels

The next charts summarize the results of extended testing focused on the impact of the amount of priority levels. The analysis was performed for several numbers of ports *n* in the range <3; 15>. For each value of *n*, 14 different numbers of bits were tested in the range <1; 20>. With each number of bits, 40 independent tests were executed with randomly generated state and optimization matrixes.

The first chart in Fig. 3 shows the relation between the number of bits used to express priority levels and the average number of iterations needed to reach the stable state for *n* = 3, 9 and 15. Only successful iteration processes, i.e. those whose final stable state fulfilled the confinement criteria, were considered.

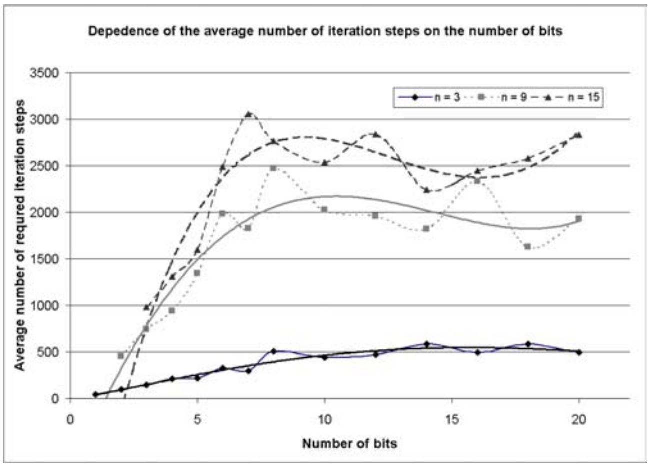

Fig. 3 Dependence of the average number of iterations on the number of bits

The second chart in Fig. 4 shows the relation between the number of bits used to express priority levels and the successfulness of the iteration process for *n* = 3, 9, 15.

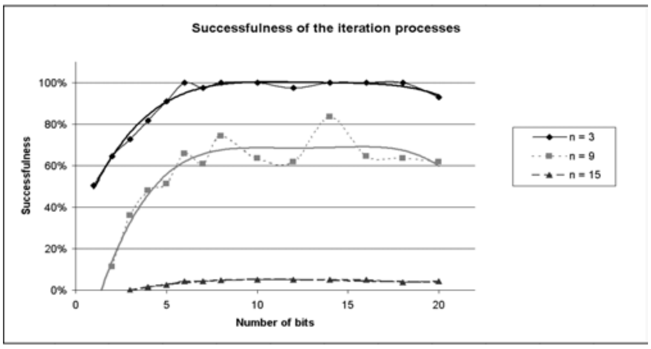

Fig. 4 Dependence of the successfulness of iteration on the number of bits

The previous charts show that by increasing the number of bits up to a certain value (7 or 8, depending on *n*) the successfulness is rapidly increasing, but still further increase has no significant effect on the successfulness. It also can be seen that for very large values of bits (about 18 or 20) the successfulness starts to decrease. On the other hand, by increasing the number of bits the average number of iterations is increased. At the beginning this rise is quite steep, but for larger values it becomes more moderate.

6 Conclusion

Based on the results of the analysis some general statements about the behaviour of the system can be formulated: As the number of nodes is increasing, the overall successfulness of processes is decreasing. Within the scope of a given number of nodes with 1 byte used to express the priority values the neural network operates effectively and the number of required iteration steps is relatively low. As the number of bits increases, the average number of iteration cycles increases. This trend culminates at about 10–14 bits and for more bits the average number of iteration cycles tends to decrease moderately.

If a single process does not reach the solution within the number of about four-five times the average number of iteration cycles, it is highly probable that the process will not converge to a solution at all.

References:

[1] Karol Molnár, Vít Vrba, Frame priorization support for switch fabrics, *3rd International Conference on Networking*, Gosier, 2004, pp. 141 - 146, ISBN 0-86341-326-9

[2] Sreeram V. Balakrishnan-Aiyer, *Solving combinatorial optimization problems using neural networks with applications in speech recognition*, University of Cambridge, United Kingdom, 1991

[3] Andrew H. Gee, *Problem solving with optimization networks*, University of Cambridge, United Kingdom, 1993

[3] Kate A. Smith, *Solving combinatorial optimization problems using neural networks*, University of Melbourne, Australia, 1996

[4] Terry Troudet, Stephen M. Walters, Neural Network Architecture for Crossbar Switch Control, *IEEE Transaction on Circuits and Systems*, vol. 38, No. 1, 1991

This paper has been supported by the Grant Agency of the Czech Republic (Grant No. 102/03/0560) and the Ministry of Education of the Czech Republic (projects No. 1K03026 and 1K04116).