Virtual Output Queuing

V. Skorpil and P. Zednicek

Abstract - A new model of the active network element controlled by neural networks has been designed. They are described the Virtual Output Queuing Manager (VOQM) blocks in this paper. They manage and group together virtual output queues. The number of queues in each block depends on the number of ports. Each VOQM has been set a unique input port number which it serves and which is used for packet addressing. The settings are made using the graphical dialog, where you can assign the already mentioned port number, the maximum number of saved packages and their maximum length.

Keywords - Network element, Neural network, Simulation, Virtual Output Queuning

I. Introduction

In the contribution the active network element is studied. The main source for the work is ref. [1] and [11]. We will be concerned with a simple switch that according to the priorities and direction of the path of input packets boots the respective packets to the output. The switch is specified by the basic transmission parameters.

- Number of lines is the number of I/O communication ports, which the given switch contains

- Transmission capacity of lines - each line has its transmission capacity in Gb/s

- Delay - time difference between the arrival of the packet, subsequent processing and its output from the switch

This research was supported by the grants:

No MSM 0021630513 Research Plan of Brno University of Technology "of Electronic communication systems and new generation of technology (ELKOM)",

No. FP-S-10-16 Research communication systems and networks, No. CZ 1.07/2.2.00/15. preparation of specialists for ICT 0139-BA Teleinformatiky,

No. CZ 1.05/2.1.00/03.0072, Centre of sensory information and communication systems (SIX)

- Loss is caused by rejection of packets due to overflow of buffers or higher traffic rules (firewall) [3]

- Throughput shows how effectively the switch can transfer input data to the output per unit of time. The ideal value is 100%

- Memory performance - each switch includes a buffer whose size differs depending on the architecture

- Processor/computing performance requirements depend on the complexity of algorithms for the management of data units in the switch

II . Model of active switching element

The model contains the basic blocks, its simplification is appropriate for the description of the activity (Fig.1). It consists of two inputs and two outputs (there exists extension for four or more inputs and outputs), two input queues and a control block with neural network. The switching area ensures switching [3].

The input and output blocks are responsible for the receipt, transmission and check of packets. The input queues are of the type of VOQ memory (Virtual Output Queuing), and in the model they act as input buffers. They are the FIFO queues, their principle consists in that the memory of each input port is divided into N virtual queues, while N is the number of ports and each queue contains only data designed for the port. It is possible to realize queues for each priority class; the number of queues is always the square of the number of ports.

The role of a control block is to decide what packet from the queue will be connected to the output port. The input queues provide the priority value of the packets that are the first in the queue at a given time [6]. From these values, a priority vector is set up, which is the input for the control neural network. The output of these blocks is the configuration vector.

The switching area provides the physical connection and the transfer of packets between the input and the output. The connection is controlled by the configuration vector. There are several types of switching areas; we will examine only the single way areas. Characteristic of the single way area is the linking of just one path between one input and output. For the four port router, there may be a maximum of four independent connections.

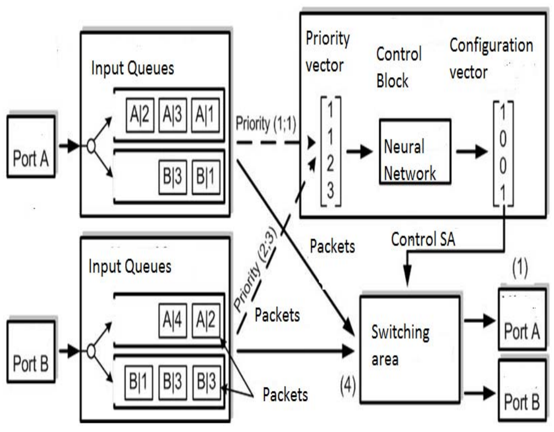

Fig. 1 Model of active network element controlled by neural network

We use one of the most widely used switching areas, the crossbar switch. The number of switching nodes is equal to N^2, where N is the number of ports. The connection between the input port i and output port j is obtained by setting the switching element (i,j) into the connection state. Setting into the connection state can be triggered for each switching element automatically by the arrival of the packet with the corresponding address output. This process can take place entirely independently of the other cells [5]. Using this solution, the control of the switching area will be simplifies considerably, and the switching area will becomes self-controlled, i.e. the control functions are distributed among the switching elements [2]. The benefits of choosing this structure can be specified as follows:

- There is no internal blocking

- The structure is simple and switching works on high frequencies

- Easy modularity and the possibility of parallel sorting multiple arrays

A. Training set, priority and configuration vectors

The priority vector contains the value of the priorities of all the packets that are on the first positions of the input queues. The values of the priorities are set in the model in the range 0-10. Zero, "0", indicates an empty queue - no packet. The priority value of "0" may not be transferred in any packet, it is designed only for internal settings. The highest priority, i.e. packet that will have precedence over the other packets, is the number "10". The lowest priority is "1". For example, the VoIP packet should would have a priority of 10, the FTP transfer a priority of 1. The size of the priority vector is given by the number of input queues.

The configuration vector is the result of the calculation of the neural network. This is the pattern that best corresponds to the given priority vector. It contains only binary values, the logical value "1" corresponds to the switched node in the switching area, and the value "0" or "-1" to the sleep disconnection node. Its size is the same as the size of the priority vector. For the Hopfield neural network the configuration vector serves at the same time as an initialization vector-learning pattern [7]. In the case of the configuration vector two unwanted states can occur. This is the emergence of conflicts and multi-port transfers. A collision occurs when in the switching area two inputs are interconnected with one output. This leads to interference and data from the two inputs will mutually interfere. In the case of multi-port transfer there will be no violation of data integrity but this situation is not implemented in the model presented because it is only a basic model.

The training set is usually a set of priority vectors and their associated configuration patterns. Finding the correct training set is often a key role when solving optimization problems. With the increasing number of ports and rules the number of possible combinations of input vectors for neural network is increasing. The dependence of the number of configuration patterns and priority vectors on the number of ports is given in Table 1. The number of the configuration patterns equals the factorial of the number of ports and thus includes only the states where only one packet can be routed to one output [4]. One packet thus cannot be routed to multiple output ports. If none of the input queues includes packets, the priority vector contains the values of the "0" priorities, the control block selects the nearest member of configuration pattern. The number of priority vectors is increasing according to the equation:

$$N = P^I, \qquad (1)$$

where P is the maximum value of the priorities, and I is the size of priority vector. As is clear from the Table, which has been calculated for P = 10, the set of input combinations of priority vectors is huge [9]. Even if the number of ports is small, working with such a large set is time consuming.

B. Algorithms for sorting and selecting

The implementation of the algorithm for queuing control is an important part of the algorithm, which is responsible for the selection of data units from the buffer and their dispatch to the switching area. These algorithms are the parts which are being replaced by neural networks. We will use mainly the following algorithms and their modifications.

- FIFO- the basic implementation of the algorithm always selects the oldest cell in the queue. A very undesirable property of the algorithm is the phenomenon of queue blocking, which occurs when the output is not free [8].

Port	Priority vector	Configuration pattern
2	10exp4	2
3	10exp9	6
4	10exp16	24
5	10exp25	120
6	10exp36	720
7	10exp49	5040

Tab. 1 The number of priority vectors and configuration patterns for 10 priority classes

In this case no cell will be transferred from the buffer of a given port, even in the event that there are other cells in the buffer which tend to free outputs [2]. This queue is easy to implement but it is the least efficient

- VOQ-(Virtual Output Queuing) – has the function of input buffers in the model

- PIM-Parallel Iterative Matching- iteration algorithm that uses the VOQ and random selection

- iSLIP – Iterative Round-Robin with SLIP- extends the capabilities of PIM [2]

C. *The frame and packet format*

The basic level of the model is in Figure 2 [1]. It consists of three main parts. The first part is used to generate and store the test packets. The second part is a model of the switch. The third part is blocks dedicated to the presentation of the measurement results.

The beginning and the end of the frame are represented by two boundary blocks "-1", "-2", which are used for the detection of the frame. The blocks between them are part of the packet. The packet contains:

-The Length of the packet and the frame. The minimum length is the number of blocks, i.e. 9 values (samples). The maximum length is physically not limited.

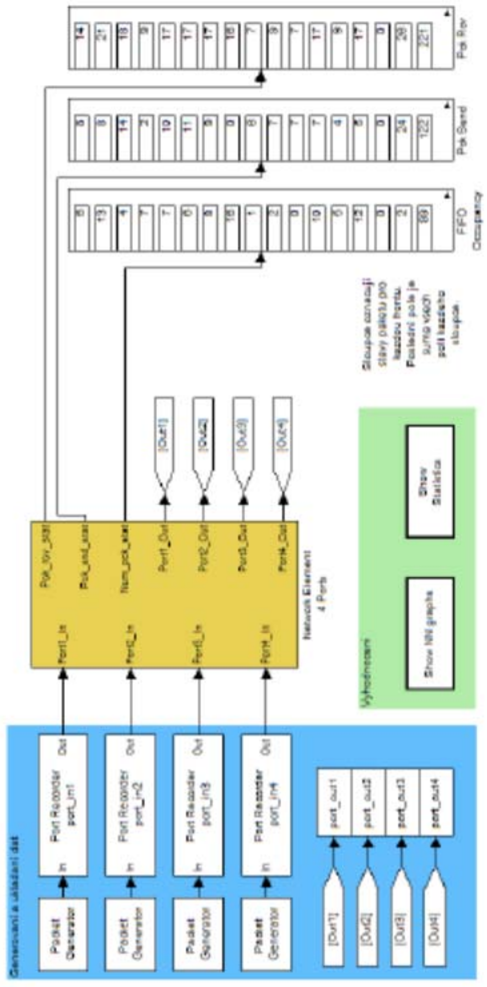

Fig. 2 Network element with testing circuits and with evaluation part

The structure of the frame and the packet are in Fig. 3 [1]

1	Length	Target	Source	Priority	Sequence	Data	Time	2

Fig. 3 Structure of the frame and packet

-The Target is the address; in our case, the value of the output vector

-The Source is the address of the port from where the packet originated

-The Priority of the packet, or also the Priority Class. The values must be in the range <1,10>, a larger number has a higher priority and therefore takes precedence over a smaller number

-Sequence is the sequence number of the packet

-Data, the content of user data, the minimum size is one sample.

- Packet dispatch time. It is used to measure the delay and jitter

D. *Packet Generator*

The Packet Generator generates packets according to a defined configuration table. It is set using the graphical form of the block - see Table 2. It is possible to set up to ten (or more) separate services with specific settings. Individual services may overlap each other as regards applications; there may even be a few identically set services, which increases the probability of service selection. The "Target", "Interval" and "Size" areas may contain a vector, i.e. a range of values [7]. The vector must have only two components, for example [1 of 4]. These two items have two values in the configuration matrix. The parameter "Size" must not contain a smaller value than "9". The parameter "Fill" is designed to fill in the user data, and its minimum value is "1". The entry "Priority" indicates the size of the priority for the packet. Its value must be between <1,10>.

The Graphics menu is created using the Matlab GUI editor. Fig. 4 [1] shows the created generating algorithm. This is based on the external sync clock "Tick", whose period corresponds to the sampling frequency of the entire model. The variable "Config" represents the configuration table of priorities, and " Data_out "is the output data port of the generator.

After starting the simulation, the "Delay" state is activated first. One of the priority classes is selected randomly and depending on the class setting in the configuration matrix packet the delay is randomly generated. After the "Wait" time, individual parts of the packet will start to be dent [5]. In the "Sequence" state the serial number of the outgoing packet is calculated. The algorithm assigns numbers according to the priorities and the target address; this information is stored in the array "Seq". Furthermore, the user data length is generated and it is sent successively. Subsequently the current state of the simulation and the last character of the frame are inserted. The cycle is repeated from the state of "Delay".

Ser vice	Prio rity	Taget	Interval	Sou rce	Si ze	Fi ll
1	1	[1 4]	[1 10]	4	9	1
2	2	[1 4]	5	4	9	2
3	3	[1 4]	7	4	9	3
…						
1 0						

Tab 2 Service settings for packet generator

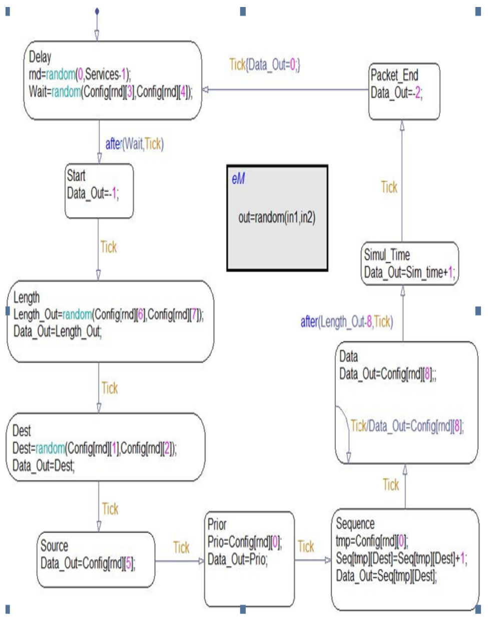

Fig. 4 Algorithm for packet generation

E. *Packet Recorder*

This block is used to store and again reproduce the data flow. The recorded flow is stored in the Matlab Workspace [8]. It is inserted following the generator of the packet. It has three working modes:

-transit mode, there no selection is marked in the settings, it is a regular connection between input and output

-saving the stream in the Workspace

-loading a data flow from the Workplace

III. VIRTUAL OUTPUT QUEUING MANAGER

The Virtual Output Queuing Manager (VOQM) is a part of the Switching Element. The block diagram of the Switching Element is shown in

Fig. 5 [1]. It is composed of the VOQ Manager, Control Element, CrossBar blocks and other utilities. This paper is focused to the VOQM.

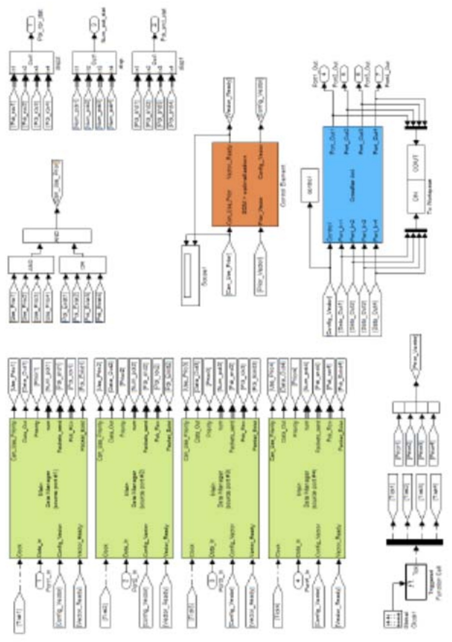

Fig. 5 Internal architecture of the network switch

The main element of the VOQM (Virtual Output Quening is the queue manager "FIFO Manager". The same as in the case of VOQM, the parameters of queues are set in the graphical window. The number of the output port is defined here that applies to the particular queue. All queue managers take over the parameters of queues of their VOQM. The internal structure is in Fig. 6 [1]. It is made up of a separate queue and of a circuit for the detection of the incoming packet. The queue is managed by three input events/functions. The first event is clock signal Tick. The second is the detection of a new incoming packet, Paket Start, and the last function is for reading data from the Pop memory [8]. The number of the queue from which reading is to start is transferred through the FIFO input.

The Queue consists of three states/diagrams ranked in parallel. Therefore, the events can take place independently of one another. They are the „Write to Queue“, „Read_from_Queue“ and the current readiness of queue (fifo state). The output variables are the current number of packets in the queue „Num_Packets“, the data output „Data Out“. „Priority“ is the priority of the first packet in the queue, the merge all the output queues forms the priority vector. „Packet_Send“ is the number of packets which were sent since the beginning of the simulation. „Packet_Receive“ is the number of packets received since the beginning of the simulation, „Pop“ describes the state of the queue; if the queue is currently sending, Pop is equal to "1", otherwise the value is "0".

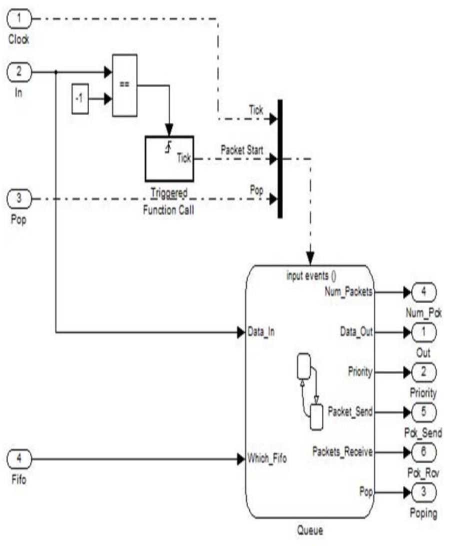

Fig. 6 Internal connection of FIFO Manager

IV. Algorithm of saving into the queue

We describe the algorithm of the packet saving into memory. An incoming packet is detected by a circuit, which invokes the activation of a transient function *Packet_start*. The condition at the same time checks if the number of the current packets *Num_Packets* in the queue is less than the maximum number of packets that can be save *Max_Packets*. If this condition is true, moves the active state to the state *Storing*. In the state *Storing* becomes to storing of the entire packet with a frame to a FIFO. When this state is activating (the arrival) the commands for writing of initiative symbol from input data variable *Data_In* are accomplished. Packets are saved by rows. The index *End* signs the current position for the storage of the packet. Additional condition checks the length of the packet. If it is greater than the maximum length, the packet is not written and the status *Idle* is activated. The condition locates whether the packet is to be stored in this queue. The value of the targets in the packet must match the port number of the output queue *My_Number*. In the event of the completion of the conditions are no longer packets stored and the status of the *Idle* is activated. If it is detected the terminating symbol of the frame, the last character is saved. Consequently, the position of the index of *End* to the maximum number of packets, and in the case of a larger index, the index is set to the first item [14], [15].

V. Algorithm of reading from the queue

The state of the *Idle* is after starting active. The output of the data variable *Data_out* is set to "zero", the index of the item to the "one" and read the row index *Start* to "zero". Read from memory runs if the transient function *Start_Popping* is activated by external functions *Pop* and the validity of the conditions is met, the queue number calculated from a configuration vector is equal to the number of the queue, and at the same time in memory is at least one packet. In this case is executed transition condition *Fifo_State* that changes the output activity in the third diagram. It makes a reading of the length of the packet to a variable *Length_out*. Packet in any of the sample is sent according to impulse *Tick*. After the last character of the frame is increased by the index of *Start*, deducted one packet of Num_Packets and increased output variable number of packets sent Packet_Receive. Check the position of the index of Start to the maximum number of packets, and in the case of a larger index, the index is set to the first item.

VI. Algorithm of a queue activity

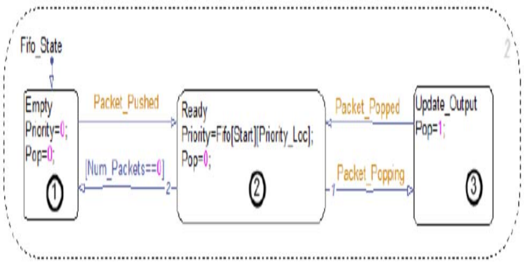

Fig.7 Algorithm of a queue activity

The algorithm of a queue activity is on a Fig.7. The start active state is again *Idle*. The queue does not contain any packet, then a queue priority *Priority* is set to system value "zero". The state *Idle* is changed to the state *Ready*, the queue is not empty and has output priority value less then "zero". The priority value of the first packet in the queue is read [12] .

VII. Control Element with Hopfield Network

Block diagram of control element with the Hopfield network is in Fig. 8 [1].

The priority vector is led to the first input of the „Switch“ block and the value of detector „Can_Use_Prior“ is conveyed to its control input is via the leading edge detector. If the variable changes the state from "0" to "1", the first input in the „Switch“ block is linked with the output, and the priority vector is brought to the input block of the neural network [6].

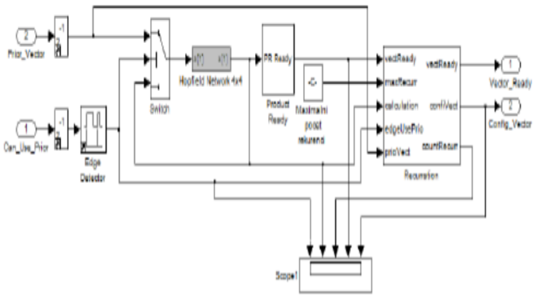

Fig. 8 Subsystem of the Hopfield network block

The neural network performs one calculation of the configuration vector and sends the result back to the third input of the "Switch" block and to the „Product Ready“ block. In the „Product Ready“ block the result and the stabilization of the network are tested, and thus whether the vector contains the values "1" and "-1". If the vector meets the condition (is correct), it is sent to the "Recurrence“ block, which covers the maximum number of recurrences. Through this block it gets to the output. If the condition is not met, the "Switch“ in the next sample is switched over to the third output (leading edge detector = "0") and the feedback of the neural network is interconnected. A recurrence connection is created. In the event of an overrun of the maximum number of recurrences, the „Recurrence“ block calculates the configuration pattern from the priority vector. The calculation is based on the distribution of the vector to quartets, from which the highest priority is then selected. It is allocated "1" and the remaining values are set to "0".

VIII. Control block SOM for a network with optimization

This block enables set a delay of sample calculation. We can see it in a Fig 9.

The basic of this subsystem is similar as for the Hopfield network. The principle of delay is, that after activation of input edge detector, time started to be calculated and after time limit exceeding the input neural network port is connected with output *Config_Vect* and *Vector_Ready* is set to "one". The neural networks in delay time calculate the configuration vector [13] .

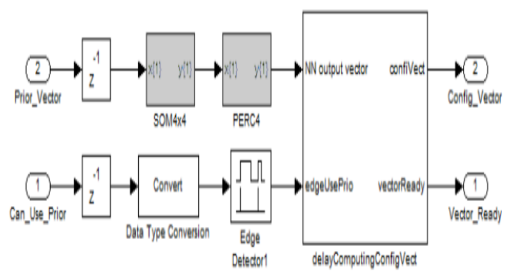

Fig. 9 Subsystem of the SOM block

IX. Neural networks

Fig. 10 [1] shows the number of incoming packets from the individual input ports to output port 3. The number of received packets is usually dependent on the size of the delay. The dependence of the delay of packets is displayed in Fig. 9 [1]. The values are obtained from the output of port 3. The x-axis shows which values of the priorities were switched over to the output port. In this case they were priorities of 1, 3, 4, 5 and 10. The diagram shows that the highest priority (10) does not have the expected minimum delay.

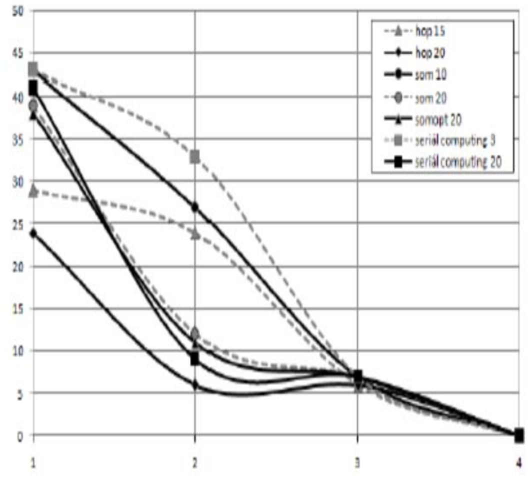

Fig. 10 Number of incoming packets (y-axis) of individual input ports to the output port (x-axis)

The aim of the work was to design and simulate an appropriate model of active network element controlled by an artificial neural network. The element has been optimized for priority switching of input requirements for the outputs. First, the theoretical model of a network element with appropriate neural networks was described. The Hopfield network, the self-assembling Kohenen maps and the Perceptron neural network were tested. These tests are described in another paper. In the case of the Hopfield network the least number of recurrences was solved for the given number of inputs of the active element. The model has been implemented in the Simulink environment and contains all the components of a network element. It appears that the best network for use studied is the classical Kohenen network without optimization; excellent results are also gives by the Hopfield network.

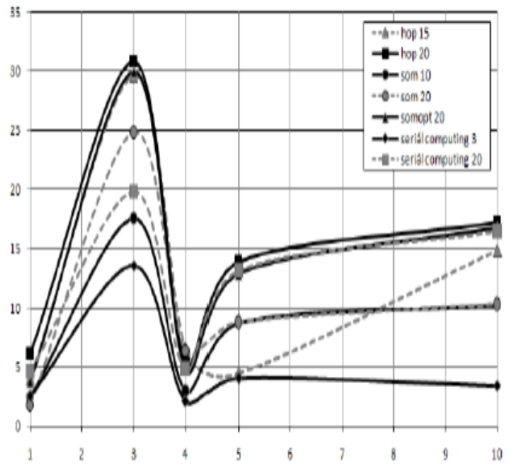

Fig. 11 Dependence of jitter [samples] (y-axis) on priority of packets [-] (x-axis) for output port 3

References

[1] P. Zednicek, *Network Element with Advanced Control*. BUT Brno, Brno 2010

[2] K. Molnar, *Hardware of Computer Networks – switching nodes*. [online], BUT Brno, Brno 2010, URL:<http://www.utko.feec.vutbr.cz/molnar/index.php?stranka=bhws>

[3] H. Demuth and M. Beale, *Neural Network Toolbox for Use with MATLAB*. Natick (USA), MathWorks, Inc., New York 1994

[4] K. Oudidi and A.H.M.Koutbi, QoS Routing Using OLSR Protocol. *Proceedings of the 14th WSEAS International Conference on Communications.* Corfu Island, Greece, WSEAS, pp. 165 - 171 Corfu 2010, ISBN: 978-960-474-200-4, ISSN: 1792-4243

[5] M. Bartl and J. Hosek and T.Matocha and K. Molnar and L.Rucka, Data Exchange Between Real Network Component and OPNET Modeler Simulation Environment. *Proceedings of the 14th WSEAS International Conference on Communications.* Corfu Island, Greece, WSEAS, pp.184-188, Corfu 2010, ISBN: 978-960-474-200-4, ISSN: 1792-4243

[6] I.Susnea and A. Filipescu and V. Minzu and G. Vasilu, Virtual Pheromones and Neural Networks Based Wheeled Mobile Robot Control. *Proceedings of the 13 th WSEAS International Conference on systems*. Rodos Island, Greece, WSEAS, pp.511 – 516, Rodos 2009, ISBN: 978-960-474-097-0, ISSN: 1790-2769

[7] I. Bogdamov and R.Mirsu and V.Tiponut, Matlab Model for Spiking Neural Networks, *Proceedings of the 13 th WSEAS International Conference on systems.* Rodos

Island, Greece, WSEAS, pp.533-537, Rodos 2009, ISBN: 978-960-474-097-0, ISSN: 1790-2769

[8] Y. Takizava and A. Fukasawa, Novel neural Network Scheme Composed of Prediction and Correlations. *Proceedings of the 13 th WSEAS International Conference on systems*. Rodos Island, Greece, WSEAS, pp.611-615, Rodos 2009, ISBN: 978-960-474-097-0, ISSN: 1790-2769

[9] The MathWorks, Inc. Stateflow® and Stateflow® Coder^{TM} 7: User´s Guide. London 2009

[10] V. Skorpil and J. Stastny, Comparison Method for Object recognition. *Proceedings of the 13 th WSEAS International Conference on systems*. Rodos Island, Greece, WSEAS, pp.607-610, Rodos 2009, ISBN: 978-960-474-097-0, ISSN: 1790-2769

[11] V. Skorpil and P. Zednicek. Model of Parts of Active Network Element. *Proceedings of the European Conference of Communications – ECCOM´10.* Tenerife, Spain, Puerto De La Cruz 2010, pp. 203-207 ISBN: 978-960-474-250-9

[12] H.Sug. An Effective Sampling Scheme for Better Multi-layer Perceptrons. *Proceedings of the 9th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and data Bases.* Book series: Artificial Inteligence Series – WSEAS. pp. 302-306, Cambridge 2010

[13] H.Sug. Performance Comparison of RBF networks and MLPs for Classification. AIC´09: *Proceedings of the 9th WSEAS International Conference on Applied Informatics and Communication – Recent Advances in Applied Information and Communication.* Book Series: Recent Advances in Computer Engineering. pp. 450-454. Moscow 2009

[14] L.P.S. Fernandez and A.R. Ruiz and J.de J.M. Juarez, “Urban Noise Permanent Monitoring and Pattern Recognition”. *Proceedings of the European Conference of Communications – ECCOM´10.* NAUN, Tenerife, Spain, Puerto De La Cruz 2010, pp. 143 - 148 ISBN: 978-960-474-250-9

[15] T.Chlouba, “Design Patterns in Mobile Architectures.” *Proceedings of the European Conference of Communications – ECCOM´10. NAUN,* Tenerife, Spain, Puerto De La Cruz 2010, pp. 286-289 ISBN: 978-960-474-250-9

V. Skorpil (WSEAS 2002)

He was born in Brno, Czech Republic, 1955. He attended the Brno University of Technology (BUT) for university education in 1980 and Ph.D. in 1989. From 1980 to 1982 he worked as a designer for the telecommunication design office. He again entered the Department of Telecommunications of BUT in 1982 as a university teacher and he has been working in this department since that time, now he is a vice-head of this department.

He takes a keen interest in modern telecommunication systems. He has taught in courses on transmission systems from analogue through all categories of digital up to special applications. He is the author of about 103 international scientific papers and some manuals. He has complemented his theoretical knowledge by co-operation with a lot of firms and institutions. He has co-operated on telecommunication projects such as digital transmission and switching systems, telecommunication broadband networks, data networks LAN and MAN, on structured cabling design, neural networks, wavelet transform, Quality of Service, data bit rate compression, etc.

Assoc. Prof. Ing. Vladislav ŠKORPIL, Ph.D. He is a member of international organisations IEEE and WSEAS.

P. Zednicek
He was a student at Brno University of Technology. He is interesting about network elements controlled by artificial neural networks. His degree is a master.
Ing. Petr Zednicek.