A BP Neural Network Based Self-tuning for QoS Support in AVB Switched Ethernet

Dong Chen and Ang Gao(✉)

Northwestern Polytechnical University, Xi'an 710027, Shaanxi, China
chendongcharlie@mail.nwpu.edu.cn, gaoang@nwpu.edu.cn

Abstract. To support QoS of time-sensitive services in Ethernet, IEEE has proposed a set of standards for transporting and forwarding real-time content over Ethernet known as Audio Video Bridging (AVB) with bandwidth reservation and priority isolation. AVB traffic is granted highest priority to ensure its transmission while low-priority traffic follows Strict Priority (SP). However, due to restrictions of SP algorithm, low-priority traffic may suffer a problem of starvation. To solve the problem, we propose a BP neural network based self-tuning controller (BPSC) over a probability selector to manage the transmission of best effort (BE) traffic in AVB switched Ethernet. This paper introduces the model of BPSC, followed by an simulation to demonstrate that BPSC could operate effectively and dynamically. The result shows that BPSC not only has the ability to manage the transmission precisely, but also shows both effectiveness and robustness.

Keywords: AVB · BP neural networks · Self-tuning
Machine learning · QoS

1 Introduction

In recent years, digital audio and video transmitted through Ethernet is increasing rapidly as the process of audio and video becomes a fundamental technology in many fields, laying a higher demand on network system. With high bandwidth, high flexibility and low cost, switched Ethernet Network has become a satisfying solution for transmission. In order to adapt audio and video data onto Ethernet Network, AVB technology was introduced. An AVB network implements a set of protocols developed by IEEE 802.1 Audio/Video Bridging Task Group (AVB TG). The core protocols include an Audio Video Bridging system (IEEE 803.2BA), timing and synchronization for time-sensitive applications (IEEE 802.1AS), stream reservation protocol (IEEE 802.1Qat) and forwarding and queuing for time-sensitive streams (IEEE 802.1Qav).

IEEE 802.1Qav standard introduced a specific mechanism to ensure the real-time audio/video transmission that is highly time-sensitive. The principle is to separate the network traffic into different classes, each with their respective

B. Li et al. (Eds.): ChinaCom 2017, LNICST 237, pp. 474–484, 2018.
https://doi.org/10.1007/978-3-319-78139-6_47

priority. There would be a portion of bandwidth reserved for AVB traffic, while other classes obey the rule of best effort transmission by appending traffic of each class into respective FIFO queue and generally being scheduled by a selector. 802.1Qav supports 2 selection algorithms in a switched Ethernet network: Credit Based Shaper (CBS) and Strict Priority (SP). CBS algorithm uses a token to manage the transmission by allowing only the queue that gets the token from the token bucket to transmit data for a limited length, while SP algorithm only transmits data from the queue that has (1) frames to transmit and (2) the highest priority among all queues that have frames to transmit at the same time.

AVB has become a research focus recently. Paper [1] purposed to use a new mechanism to reduce the guard band by accurately calculating the size of frame that can preempted. Paper [2–4] worked on analysis of worst-case scenarios of AVB network for multiple applications. However, most of these papers focused on resource reservation and the transmission of AVB traffic, while BE traffic not taken into consideration. Typically, SP algorithm is used for BE transmission, and networks exploiting SP are very likely to encounter starvation of low-priority traffic in which traffics with very low priority are not granted enough resources for transmission while high-priority traffics are allocated too much resources. In practice, there does exist situations that low-priority frames has little chance to transmit because high-priority queues have sustaining frames to transmit.

In this paper, we propose a probability selection model for non-AVB classes transmission based on a BP neural network controller to replace typical SP algorithm. This BP neural network based controller shows a ability to adjust the system dynamically and robustness for different traffic environment. It enables high-priority frames to transmit with less delay, while low-priority frames still have a chance to send so that over-sacrifice of them can be prevented.

2 BPSC Architecture

2.1 System Modeling

Suppose there are I classes of BE traffics in AVB switched Ethernet, each with a priority parameter δ set by upper layer protocol indicating the priority. The goal of the controller is to maintain all parameters to meet Eq. 1:

$$\frac{\zeta_i}{\zeta_j} = \frac{\delta_i}{\delta_j}, i, j = 1, 2, \ldots, I \tag{1}$$

where ζ_i is the measured average queuing delay of class i. Apparently, class with a smaller δ would expect a lower delay and thus a higher priority. Moreover, the proportional relationships can be simplified by setting ζ_1 as the standardization factor, thereby we can obtain the condition illustrated in Eq. 2:

$$y_i = \frac{\zeta_i}{\zeta_1} = \frac{\delta_i}{\delta_1}, i = 2, 3, \ldots, I \tag{2}$$

where y_i is defined as desired relative index (RI) for each class to maintain. Note that this equation illustrates that there are $I-1$ independent constraints for this system. As for the selector, we define the probability of transmission p_i for class i at the k^{th} sample period. The probabilities of all class obey the constraint in Eq. 3:

$$\sum_{i=1}^{I} p_i(k) = 1 \text{ for all } k \tag{3}$$

This enables frames in high-possibility classes to have a relative small delay, while frame in low-priority classes still have a chance to transmit as long as $p_i(k) > 0$. Due to nonlinearity between selector probability and the delay, the probability cannot be settled based on desired relative index only: frame arriving rate and the throughput should be considered.

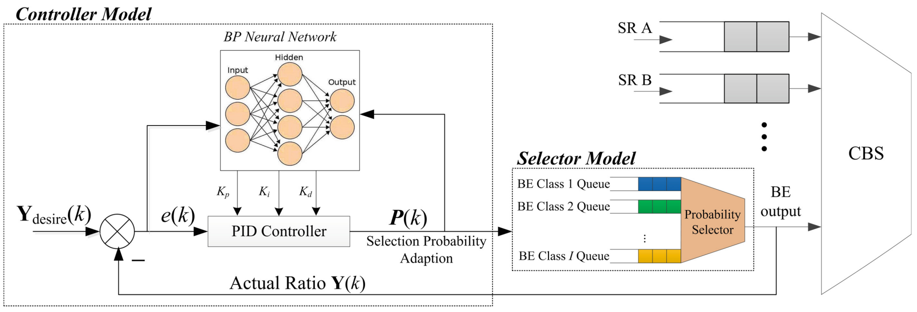

Fig. 1. System model of BPSC

Figure 1 shows our model of BPSC. It is composed of 2 parts: a selector model and a controller model. As for the selector, the input is a probability vector with $I - 1$ independent variables used for next sample period while the output is the measured RI, $\mathbf{Y}(k)$, of this sample period. This measured RI, together with the preset desired RI, $\mathbf{Y}_{\text{desire}}(k)$, are sent into the controller as its input. The controller will adjust its BP neural network accordingly so that all parameters will automatically change to a suitable value. And then, controller computes $P(k)$, the update of selection probability for next sample period, and sends it to selector.

By implementing this model, the measured RI is expected to adjust dynamically in accordance with frame arriving rate, and such a feedback mechanism could maintain the stability of the system so that the uncertainty of traffic flow can be well handled.

2.2 Core of BPSC: BP Neural Network

Artificial Neural network (ANN or NN) has been widely used in many disciplines including robotics, recognition and computer vision. It is a computational model simulating human being's neural network to solve problems in the similar way as a human.expressrelationship because of its nonlinearity. The basic element of

NN is the single recurrent neuron which adds all its input with different and variant weight, as well as the threshold, as its primary output:

$$S_j(k) = \frac{1}{2}\sum_{i=1}^{N} w_i x_i - \theta_j \tag{4}$$

and then the $S_j(k)$ is taken to the activation function, the result being exported:

$$f_j(k) = f(S_j(k)) \tag{5}$$

In Eqs. 4 and 5, j denotes the label of the neuron, N is the total number of neuron's input, w is the weight foreach input, θ is the threshold and f is the activation function of the neuron. The common activation functions in practice include sigmoid, hyper tangent and Rectified Liner Units (ReLu) (Fig. 2).

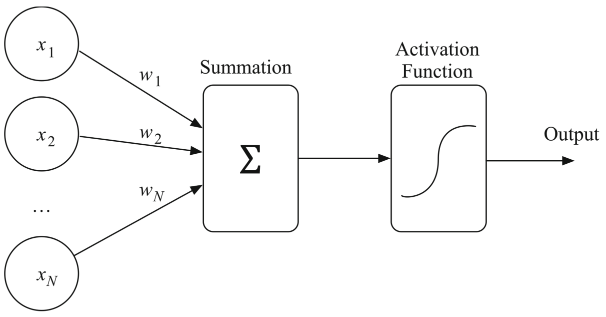

Fig. 2. The model of a neuron

A simpler representation of Eq. 4 is to combine inputs and delayed outputs together as the input of a single neuron, i.e.,

$$S(k) = \mathbf{W}^T\mathbf{X} \tag{6}$$

where

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

With multiple neurons in different layer, a network can be established. In a typical neural network, an input layer and a output layer are necessary. Each layer would contain one or more neurons, which enables parallel computing. A simple way of supervised learning to use NN is to feed the network with abundant datum, each with a label indicating the desired output. Then the NN uses optimization algorithm to approach the labels as close as possible. This whole process is called training. After training (and validation, in most cases, to check if NN after training operates as we expect), all parameters in the NN will be settled and we can feed the NN with raw data and use its output for further work. However, in real network environment, most channel parameters such as channel quality and speed rate are time-varying and thus a pre-trained NN cannot fully satisfy our requirement.

BP neural network is a commonly used algorithm based on error back propagation algorithm of multilayer feedforward neural network. A typical BP neural network contains not only the input and output layer, but one or more hidden layer to process the data. The whole process starts from the input layer that spread the input data to hidden layer, then the hidden layer would put the summation from input layer to the activation function, and finally the output layer would collect the output of hidden layer to form the output of the system. Though there is no upper limit of layers, in this paper we propose to use only 3 layers, the first and the last being the input and output layer respectively, while second one being the hidden layer. This is because a 3-layer neural network is able to learn any function theoretically [6], and the computation amount would increase exponentially as more hidden layers are added. The BP neural network implemented in our system is shown in Fig. 3.

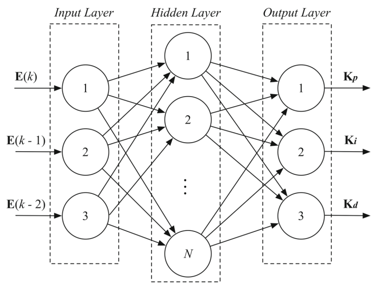

Fig. 3. BP neural network model

$\mathbf{E}(k)$ is the error between the desired RI and the measured RI of class i to class 1, i.e.,

$$\mathbf{E}(k) = \mathbf{Y}(k) - \mathbf{Y}_{\text{desire}}(k) \tag{7}$$

$\mathbf{K}_p, \mathbf{K}_i, \mathbf{K}_d$ are self-tuning coefficient vectors with $I-1$ elements each, and N is the number of neurons in hidden layer.

The BP neural network calculates the self-tuning coefficients based on pre-trained weights, and at the same time, is trained with data from last sample period by back propagating the error to weights. Therefore, the network is able to handle a system with dynamic traffic arriving rate by adapting the selector accordingly.

Assume that the numbers of neurons in these layer are n_1,n_2 and n_3 respectively. When an example used for training is obtained, the first process is forwarding, which requires 2 matrix multiplications, and the computation amount is $n_1 * n_2 + n_2 * n_3$. Since n_1 and n_3 are determined by the problem while n_2 is set by users, the time complexity is $O(n^2)$ given that time complexity for each neuron is $O(1)$. The second process, back propagation, has a similar computation amount as forwarding. Therefore, the time complexity of training BP neural network is $O(n_2)$.

2.3 Model Algorithm

Table 1 shows the process of complete BE transmission system, including BPSC and possibility selector model.

Table 1. Algorithm of selector with BPSC

Step 1	Initialize: $P(0) \leftarrow [\frac{1}{I}, \frac{1}{I}, \ldots, \frac{1}{I}]^T$; $w_{ij} \leftarrow$ for all $i, j; k \leftarrow 0$
Step 2	Set $\mathbf{Y}_{\text{desire}}$ as required and learning rate η for NN
Step 3	Check the queues of all classes, use an indicator vector $\mathbf{A}$ to record if there is traffic, i.e., $A_i = \begin{cases} 0 & \text{if class } i \text{ has frame(s)} \\ 1 & \text{if class } i \text{ has no frame} \end{cases}$
Step 4	$M \leftarrow \mathbf{A}^T\mathbf{P}$
Step 5	Generate a pseudorandom number m that obeys uniform distribution in the range of $(0, M)$
Step 6	Decide that frame in class i to transmit if $A(i) = 1$ and $m \in (p_{i \text{ lower}}, p_{i \text{ upper}})$, where $p_{i \text{ lower}} = \sum_{j=1}^{i-1} p_j$ and $p_{i \text{ upper}} = \sum_{j=1}^{i} p_j$
Step 7	Transmit the frame selected in step 6 and collect its delay: $D_i = D_i + d$, $R_i = R_i + 1$ where i is the class of frame, $\mathbf{D}$ is a vector that saves the summation of each class's delay, d is the delay of the frame transmitted and $\mathbf{R}$ is a vector that saves the total number of frames transmitted in this sample period
Step 8	After transmission, if current sample period is not expired, go to step 3, else goto step 9
Step 9	$\zeta_i(k) \leftarrow \frac{D_i(k)}{R_i(k)}, Y_i(k) \leftarrow \frac{\zeta_i(k)}{\zeta_i(k)}$
Step 10	Feed NN with $\mathbf{Y}(k), \mathbf{Y}_{\text{desire}}(k)$, $\mathbf{E}(k) \leftarrow \mathbf{Y}_{\text{desire}}(k) - \mathbf{Y}(k)$
Step 11	NN forward: obtain self-tuning parameters with NN
Step 12	Update possibility vector $\mathbf{P}(k+1)$ with self-tuning parameters
Step 13	NN error back propagation: update w_{ij} in negative gradient direction with learning rate η
Step 14	$k \leftarrow k + 1$
Step 15	Go to step 3 and continue transmission

3 Experiments and Results

3.1 Configuration

An software simulation is taken to test our BPSC for frame forwarding in AVB switched Ethernet. In our simulation, the total output bandwidth of switcher is 50 Mbps, 60% of which is allocated for frame transmission of AVB frames. Besides AVB traffics, there are other 4 kinds of BE traffics marked class 1, 2, 3 and 4 in the order of descending priority. Due to the fact that bandwidth reservation strictly isolates the transmission of AVB and BE, the following analysis focuses on the transmission of BE.

The interval of each frame arriving at the queue for each class obeys Gaussian distribution with $\sigma_i = 1/v_i$, where i is the label of class and v denotes the expected frame arriving rate. Meanwhile, the length of each frame obeys Pareto distribution with an average of 400 bytes.

In our simulation, we set $\mathbf{Y}_{\text{desire}}(k) = [1.3, 1.4, 1.7]^T$ for all k and we run dynamic and static test separately. By running a dynamic test we would like to observe how BPSC deals with burst change of communication environment; with a static test we hope to see if BPSC could manage the transmission to meet our requirement, i.e., desired RI.

3.2 Dynamic Performance

To illustrate how BPSC operates in changing environment, and to make the comparison between BPSC with existing algorithm, we exploit an off-and-then-on mechanism by dividing the whole simulation into 5 consecutive stages, in which the details of each stage are listed in Table 2.

Table 2. Simulation details of 5 stages

Stage	Arriving rate of class 1 (fps)	Arriving rate of class 4 (fps)	Arriving rate of class 2 & 3 (fps)	Selector algorithm	Controller online or not	Length of this stage (second)
1	1100	1100	1100	SP	N/A	100
2	1100	1100	1100	Possibility selector	N	100
3	1100	1100	1100	Possibility selector	Y	100
4	1600	600	1100	Possibility selector	Y	100
5	600	1600	1100	Possibility selector	Y	100

The frame arriving rate at the queue is shown in Fig. 4(a), while Fig. 4(b) and (c) shows the mean delay and RI of each class respectively.

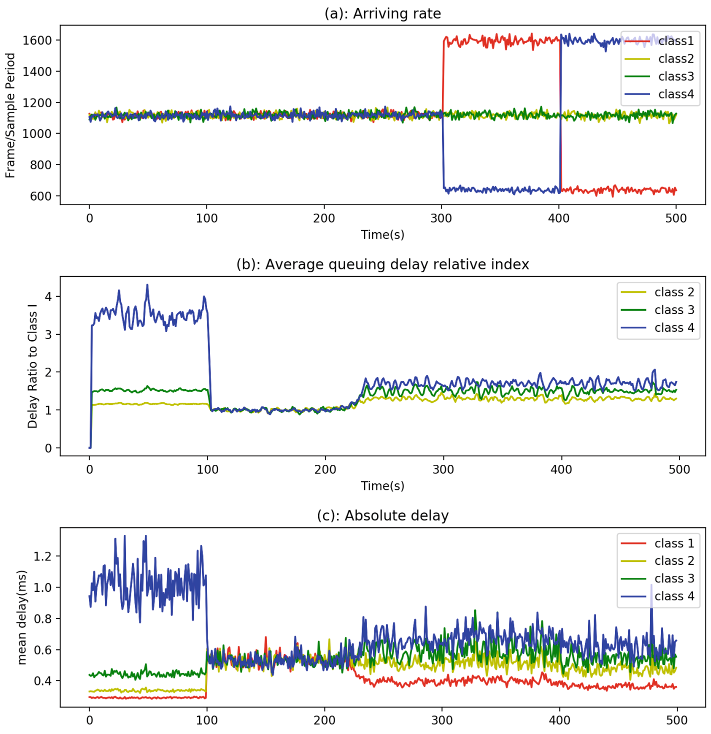

Fig. 4. Result of dynamic test

1. In the first stage (0–99 s), the mean delay of class 1 to 3 traffic is much shorter than class 4 traffic. This indicates that SP algorithm, though it could manage the transmission of traffic with different priority, may have the problem of over-sacrifice.
2. In the second stage (100–199 s) when possibility model is applied, the mean delay of each class traffic quickly stabilized and the measured RI is fluctuating around 1, showing that our possibility selection model could quickly stabilize the system.
3. Our controller is launched in the third stage (200–299 s), after 100 s of simulation. The controller takes about 20–30 s before its stabilization. After that, the measured RIs fluctuate slightly around the desired RI due to randomness of traffic. However, compared with first stage, our controller well manages the fluctuation in a smaller degree and maintains desired RIs.
4. In the fourth stage (300–399 s), the increase of frames in class 1 and decrease in class 4 make RI of class 2, 3 and 4 go up, especially class 4. A big fluctuation at the beginning of stage 2. But this situation settled very quickly. Similarly, in the last stage (400–499 s) all classes of traffic suffered an drop of RIs but being controlled quickly.

During the whole process, contingency of the length and interval of frames may disturb our system but our system shows robustness by adjusting itself dynamically and controlling the probability accordingly. Furthermore, by comparing the first and third stage, we can observe a better performance due to the implementation of our controller in respective of fluctuation, absolute delay and RI.

3.3 Static Performance

To evaluate the static performance of our controller, we collect the data of delay when the system turns stable with varying arriving rate which changes between 1100 and 1500 fps for each class. We first run the simulation using SP algorithm, then BPSC. The result of each controller is shown in Figs. 5 and 6 respectively.

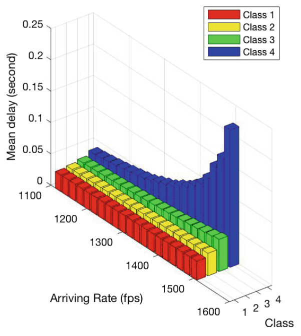

Fig. 5. Relation of delay and frame arriving rate using SP algorithm

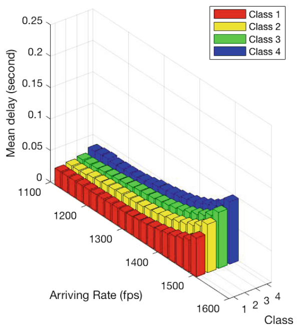

Fig. 6. Relation of delay and frame arriving rate using BPSC

1. As we can see in Fig. 5, when we are using SP algorithm for traffic management, it is inevitable that low-priority traffic has the problem of starvation as arriving rate increases while traffic with high priority barely increase their delay.
2. As shown in Fig. 6, when arriving rate is relatively low, system using BPSC cannot fulfill the desired RI. This is because the low arriving rate creates a situation such that frames arriving at the queue would be transmitted immediately: in most cases there is no other frames being transmitted.
3. However, frame in low-priority class still has a higher mean delay, and as the arriving rate increases, delays of 4 classes all increase at different speeds, suggesting that BPSC could manage to adjust the system to work as expected.
4. Since congestion control is not covered in this paper, the output data rate should always be larger than the frame arriving rate, which restricts the growth of input data rate. Nevertheless, this comparison illustrate that BPSC could control the traffic precisely without wasting throughput.

4 Conclusion

Our paper provides a new approach to support QoS for BE traffic in AVB switched ethernet by exploiting a BP neural network based self-tuning controller. As AVB technology developing rapidly, it is expected to be implemented in typical Ethernet to enable further management of traffic control. The controller we purpose in this paper, BPSC, is capable to manage the BE traffic in AVB switched ethernet. It could not only provide QoS service, but also maintain mean delay of each class that satisfies desired RI without reduction of throughput. Furthermore, in our simulation, BPSC shows its effectiveness and robustness, as well as the ability to prevent over-sacrifice problem that commonly happens using SP algorithm. Therefore, such BPSC has a great advantage over typical SP algorithm and could be implemented for BE traffic management.

Acknowledgments. This work was supported by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2016JM6062, in part by the Aerospace Science and Technology Innovation Fund of China Aerospace Science and Technology Corporation, and in part by the Shanghai Aerospace Science and Technology Innovation Fund under Grant SAST2016034 and the China Fundamental Research Funds for the Central Universities under Grant No. 3102017ZY029.

References

1. Wang, Y., Shen, H., Duan, D.: On stabilization of quantized sampled-data neural-network-based control systems. IEEE Trans. Cybern. **PP**(99), 1–12 (2017)
2. Cao, J., Cuijpers, P.J.L., Bril, R.J., Lukkien, J.J.: Tight worst-case response-time analysis for ethernet AVB using eligible intervals. In: 12th IEEE World Conference on Factory Communication Systems, WFCS 2016, 3–6 May 2016, Aveiro, Portugal (2016)

3. Park, J.-D., Cheoun, B.-M., Jeon, J.-W.: Worst-case analysis of ethernet AVB in automotive system. In: 2015 IEEE International Conference on Information and Automation, ICIA 2015 - In Conjunction with 2015 IEEE International Conference on Automation and Logistics, 8–10 August 2015, Yunnan, China (2015)
4. Diemer, J., Rox, J., Ernst, R., Chen, F., Kremer, K.-T., Richter, K.: Exploring the worst-case timing of ethernet AVB for industrial applications. In: 38th Annual Conference on IEEE Industrial Electronics Society, IECON 2012, 25–28 October 2012, Montreal, QC, Canada (2012)
5. Liang, J., Qu, Y.: Intelligent Control Technology. Harbin Institute of Technology Press, Harbin (2016)
6. Funahashi, K.: On the approximate realization of continuous mappings by neural networks. Neural Netw. **2**(3), 183–192 (1989)
7. Gao, A., Hu, Y., Li, L., Li, X.: A feedback based probability selection for frame forwarding in AVB switched ethernet. In: 4th International Conference on Information Systems and Computing Technology (2016)
8. IEEE Standard for Local and Metropolitan Area Networks-Timing and Synchronization for Time- Sensitive Applications in Bridged Local Area Networks, IEEE Std 802.1As, November 2010
9. IEEE Standard for Local and Metropolitan Area Networks, Virtual Bridged Local Area Networks, Amendment 14, IEEE Std 802.1Qat, September 2010
10. IEEE Standard for Local and Metropolitan Area Networks, Virtual Bridged Local AreaNetworks, Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams, IEEE Std 802.1Qav, January 2010
11. Diemer, J., Rox, J., Ernst, R.: Modeling of ethernet AVB networks for worst-case timing analysis. Math. Model. **45**(1), 848–853 (2012)
12. Zhang, X.Y., Gao, P.J., Liu, Y.: The researching and simulation of BP neural network PID controller in industrys control system. Tech. Autom. Appl. **37**, 9–12 (2010)
13. Diemer, J., Thiele, D., Ernst, R.: Formal worst-case timing analysis of ethernet topologies with strict-priority and AVB switching. In: International Symposium on Industrial Embedded Systems (2012)
14. Lim, H.T., Herrscher, D., Chaari, F.: Performance comparison of IEEE 802.1Q and IEEE 802.1 AVB in an ethernet-based in-vehicle network. In: International Conference on Computing Technology and Information Management (2012)
15. Alhowaide, A.A.Z., Doulat, A.S., Khamayseh, Y.M.: Performance evaluation of different scheduling algorithms in WiMAX. Int. J. Comput. Sci. Eng. Appl. **1**(5), 81 (2011)
16. Lee, H., Lee, J., Park, C., Park, S.: Time-aware preemption to enhance the performance of audio/video bridging (AVB) in IEEE 802.1 TSN. In: IEEE International Conference on Computer Communication and the Internet. IEEE (2016)
17. Wang, T., Gao, H., Qiu, J.: A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Trans. Neural Netw. Learn. Syst. **27**(2), 416–425 (2016)