Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Studying The Effect of Sampling Time and Network Load on Wireless Networked Control Systems

Russul N. Abdul-Hussain [a] **, *Osama A. Awad*** [b]

[a] *Department of Networks Engineering , Al-Nahrain University, Baghad , Iraq.Email: rna4mhm@gmail.com*

[b] *Department of Systems Engineering, Al-Nahrain Engineering ,Baghdad , Iraq.Email: : usamawad@gmail.com*

ARTICLE INFO

Article history:
Received: 17 /04/2019
Rrevised form: 00 /00/0000
Accepted : 19 /05/2019
Available online: 01 /09/2019

Keywords:

WNCS, PID, FOPID, PSO, Time delay, Number of nodes

ABSTRACT

Wireless networked control systems (WNCS) are considered one of the main attracting topics in the control, communication and computing domains. One of the main problems that face these systems is the various time delays which occur in the measurements. The control design process would be complicated if there is a time delay in the feedback loop. Moreover, this delay affects the system performance in terms of instability. The literature of WNCS shows that the available controllers can deal with a limited number of nodes. Beyond this limit, the system becomes unstable. In this paper, an optimized Particle Swarm Optimization (PSO)-based PID and FOPID controllers are proposed for WNCS to control the plant wirelessly. The proposed controllers aim at minimizing the time delays and can make the system remains stable when the network tackles a large number of nodes with variable length packet sizes. The performances of the proposed optimized controllers are compared to show which one of them can be more robust against the challenges of WNCS, namely varying time-delays, to handle a greater number of nodes. The results show that the FOPID is more powerful than the PID controller.

MSC.

1 . Introduction

Wireless communication networks witness several technological developments, especially on those which support online control. Wireless Networked Control Systems (WNCS) are preferred over the traditional point-to-point

Corresponding author Russul N. Abdul-Hussain

Email addresses: rna4mhm@gmail.com

Communicated by Osama Ali Awad

control systems for many reasons, such as inexpensive, flexible, mobile, easy to install and scalable, just to name a few [1]. However, the introduction of wireless networks into a control system emerges new problems, one example includes network time delays [2].

A Proportional–Integral–Derivative (PID) controller, also called a three-term controller, is one of the oldest and simplest control strategies, which has been successfully employed in the industrial control domain. In addition to its simplicity of design, this type of controllers is characterized by high-quality performance in terms of low percentage overshoot. Furthermore, for slow industrial processes, PID requires a small settling time [3,4]. It has been widely used in the industrial control field because of its simplicity of design, good performance including low percentage overshoot and small settling time for slow industrial processes [3,4]. However, the performance of a conventional PID controller can be improved by using fractional orders for I and D, resulting in what is known as a Fractional Order Proportional–Integral–Derivative (FOPID). Besides the conventional three-term parameters (K_p, K_i, and K_d) of the PID controller, the FOPID controller contains two additional parameters, namely λ and μ. These five parameters make the FOPID more flexible [5-7].

The performance of a controller is greatly depending on the values of its parameters. Therefore, finding optimal values for K_p, K_i, K_d in case of PID, and K_p, K_i, K_d, λ, and μ in case of FOPID would result in getting an optimal controller. Hence, the problem is to select a powerful search technique that can find the optimal values for the parameters. There are many search algorithms available in the literature. One of these algorithms is the Particle Swarm Optimization (PSO), a population-based meta-heuristic search method which mimics the social behavior of fish schooling or bird flocking. PSO is featured by its simplicity, flexibility and straightforward implementation, and hence, it has been applied successfully for various optimization problems [8], [9]. In this paper, two PSO-based controllers, namely PSO-PID and PSO-FOPID controllers, are proposed to control the plant over a wireless communication network.

The rest of this paper is structured as follows. Section II presents related work. Section III explains the TrueTime toolbox while section IV presents the Simulation of WNCS. The proposed PSO-PID and PSO-FOPID Controllers are introduced in Section V. The experimental results of applying the proposed methods are discussed in Section VI. Finally, the conclusions and future work are provided in Section VII.

2. Related Work

The authors in [10] used TrueTime to investigate the plant response and to study the impacts of packets delay caused by the wireless sensor network (802.11b). The concept of packet priority was used to enhance the motor response. The experimental results show that the steady state time of the actuator is significantly raised by a wireless network, especially when the wireless channel is loaded. Moreover, they concluded that the sensor packets delay has more impact.

A study presented by [11] proposed a new Fuzzy-PID controller for Ethernet 802.3 Network Control System (NCS). The authors investigated the time delay of the conventional PID controller to design the new controller, which aims at handling the time delays in NCS. The aspects of fuzzy and gain scheduling control were utilized to measure the actual time delay in NCS. An NCS model was built to analyze the performance of the proposed controller. The experimental results show that Fuzzy-PID controller achieves impressive results with a time-varying delay.

The authors in [12] designed two controllers, which are the Sliding Mode and PID, to control the speed of a DC-motor over ZigBee network, which is an NCS-based communication network. Their study investigates the stability of the overall system by analyzing the performance for some boundaries of the problems that may occur in a specified wireless NCS. The authors used the TrueTime simulator under MATLAB to simulate various network scenarios for the two controllers by considering different ZigBee network parameters. The achieved results show that the data rate of the ZigBee network is significantly affecting the speed of the DC-motor, in which the performance of the Sliding Mode Control outperforms the performance of the PID controller in terms of the data rate change.

A work presented by [13] assessed the performance of ZigBee and Wi-Fi protocols in terms of power control scheme to help WSN planning. The work also analyzed the effect of interference in the network. The authors used the TrueTime simulation platform in their assessment. From using power control scheme, the experimental results

show that Wi-Fi has high signal reach capabilities then ZigBee. Moreover, the authors observed that Wi-Fi has less effect of interference compared to ZigBee.

The authors in [14] suggested a model that simulates the multi-fingered robot hand, for which they proposed a PSO-based PID controller. The performance of the proposed PSO-PID was compared with some selected conventional and modern controllers, in which the output responses were analyzed. The obtained results show the proposed controller achieved better results than the other selected controllers in terms of system parameters. These parameters include settling time steady state error, maximum overshoot and rising time.

3. TrueTime Toolbox

To provide a platform in which embedded and distributed control systems can be designed and simulated, a MATLAB-based simulation toolbox, called TrueTime, was developed at Department of Automatic Control, Lund University, Sweden in 1999. The main blocks in this toolbox library (version 2.0, 06-04-2016), which are shown in fig. 1 [15], include Kernel, Network (wired network), Send, Receive, Battery, Wireless Network, and Ultrasound Network. These blocks are discrete, variable-step and MATLAB S-functions written in C++.

To simulates a computer node, the Kernel block is used. This block can be initialized by calling some functions written using C++ programming language or in m-file. Similar to a real-time operating system kernel, the TrueTime Kernel block is responsible for executing user-defined tasks and dealing with interrupts handlers, which include: input/output tasks, network interfaces, control algorithms, and timers. The A/D input and D/A output ports can be used to allow the kernel block to communicate with other MATLAB blocks. Three types of scheduling policies are supported by TrueTime Kernel, namely Fixed-Priority scheduling (prioFP), Deadline-Monotonic scheduling (prioDM) and Earliest-Deadline-First scheduling (prioEDF). To sort the necessary data for each task, every Kernel block is connected to an independent local memory.

On the other hand, TrueTime Network and Wireless Network blocks are responsible for distributing messages between computer nodes according to a selected network model. Seven network protocols are supported by TrueTime Network block, more precisely Switched Ethernet, FlexRay, Round Robin, Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), Carrier Sense Multiple Access/Arbitration Message Priority (CSMA/AMP) and Carrier Sense Multiple Access/Collision Detection (CSMA/CD). Similarly, TrueTime Wireless Network block operates exactly like the wired one, however, it specifies the x and y inputs to determine the location of the nodes. Two network protocols are supported by TrueTime Wireless Network block, namely ZigBee (IEEE 802.15.4) and WLAN (IEEE 802.11b/g).

To establish a WNCS using the TrueTime toolbox, the following steps are required:

1. Construct a real-time WNCS model using TrueTime and MATLAB/Simulink.
2. Design of nodes by configuring the TrueTime Kernel block to build each node, such as controller and sensor.
3. Select appropriate parameters and use the m-file code functions to set some initial values for all nodes.
4. Design of network by determining a network type, number of nodes, transmit power, data rate, minimum frame and packet loss probability, etc.

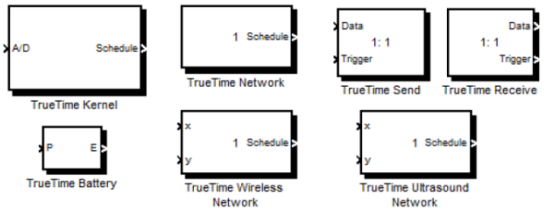

Figure 1. TrueTime Toolbox Library

4. Simulation of WNCS

WNCS demands software to analyze and simulate the timing effects on control performance [16]. The attendance of WNCS simulation is to simultaneously simulate both the system dynamics and network events. Co-simulation of a wireless network with a control system has been implemented in MATLAB/SIMULINK (TrueTime). Many simulations have been used for analysis WNCS, the reason behind using TrueTime in control system design is discussed in [10]. Wireless local area network (WLAN 802.11b) has recently used in real-time contexts. Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) is employed in this wireless technology. The standard frame format for 802.11b is shown in figure 2, the minimum frame size according to the standard description is 272 bit. WLAN has x and y inputs and these two inputs are existing in the block of Wireless Network to define the positions of the nodes (Controller node, Sensor/Actuator node, and Intereference_sender and Interference_receiver nodes). Controller node (*Xc,Yc*) location is positioned at (60,0) , for Sensor/Actuator node (*Xsa,Ysa*) location is positioned at (0,0) and for Interference_sender (*XIs,YIs*) location is positioned at (1,0) and for Interference_receiver(*XIr,YIr*) location is positioned at (5,0). The locations of interferences nodes to the sensor/actuator node are set according to the experiment studied in [17]. All the nodes are connected wirelessly through 802.11b technology at a data rate (11 Mbits/s), figure 2 shows the default parameters value of 802.11b. WNCS that had been designed, in Truetime, to study the performance of the control system over a wireless network.

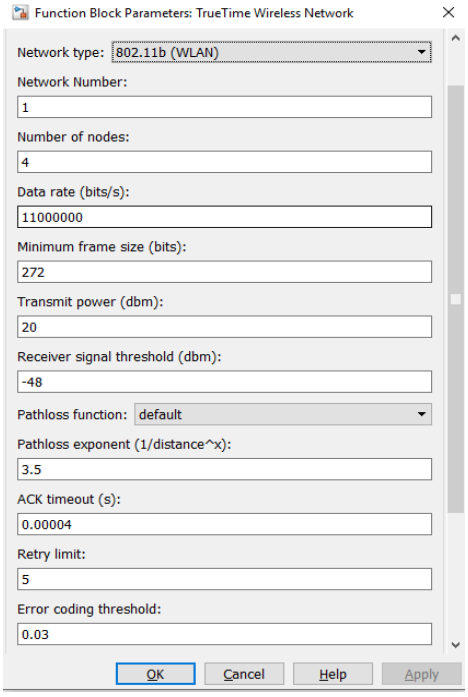

Figure 2. Parameters Value for 802.11b Wireless Protocol

Simulation time has been assumed to be 20 minutes. Where the local side consists of a TrueTime Kernel (sensor/actuator) node connected directly with the plant via A/D and D/A ports. The scheduling policy for this kernel is chosen as "prioDM". This kernel implements two tasks: sensor and actuator tasks, both share the same local memory storage. The sensor task is a time-based mode of operation, i.e. sensor will sense the plant every directly from A/D input port, preparing sensor_packet with value of readable output of plant, its type named as 'sensor_signal', plant measurement and the measured time delay or RTT in the data field of the packet, sending the sensor_packet to controller node over network and starting timer when packet is sent. The actuator task is responsible for doing D/A conversion of the control signal to actuate the plant with the newly received value. The actuator task is an event-based mode of operation. In other words, it will Read or pull the

controller_packet from network interface named as 'control_signal', stop the timer and save it in an array named as RTT and do D/A conversation of the control signal to actuate the plant with the newly received value.

The remote side consists of a TrueTime Kernel (Controller node) connected directly with a reference or setpoint input via A/D port. The reference input is chosen as a unit step with an amplitude of 1v. The scheduling policy for this kernel is chosen as „prioFP. This kernel implements the controller task which is either implement the PSO –PID controller algorithm or PSO-FOPID controller algorithm. The controller task is an event-based mode of operation. In controller node, When the task of controller_task is triggered, first, it receives sensor_signal from the network interface, then this signal will be processed by PID or FOPID controller and this signal will be sent as control_signal to sensor/actuator node. The triggering of the controller task will execute the following steps sequentially:

1. Reading the sensor_packet from the network interface.
2. Implementing the desired controller algorithm for producing a control signal.
3. Reading the control values from A/D port.
4. Preparing the controller_packet. This packet is designed to contain: Values of the controller and the name of the signal as 'control_signal'.
5. Sending the controller_packet to actuator node over the network.

Two TrueTime Kernel blocks (Interference_sender and Interference_receiver nodes) are used. Interference_sender node is designed to send packets over network with different length (The proposed idea of Interference node is approximately similar to the idea based on [18]) and hence a time-varying delay will be induced in the communication channel and Interference receiver node is designed to receive the number of packets from Interference sender node to make sure that the network is loaded with certain number of packets from the emulated nodes by interference sender node, study the effect of this load on system performance and see how many numbers of nodes can network handle. Four parameters define the operation mode for interference sender node:

1. *maxP*: The Maximum packet size.

2. *BWshare*: a fraction of the occupied bandwidth by this node.

3. *w*: weighting index for the parameter *BWshare*, w is a random number between 0 and 1.

4. nodes: number of computer nodes emulated by the interference sender node, so Interference sender node will send a packet with size: nodes x *maxP* × *BWshare* × *w*.

Obviously, when a high value of BWshare is used, high network traffic will be generated and vice versa.

The controller node uses an optimized PID and FOPID control algorithm by PSO to control the coordinated motor, the control signal is sent back to the sensor/actuator to actuate the motor. The Stepper motor transfer function and all its relevant parameters are adopted from [15], the transfer function of the stepper motor is shown in (1):

$$G(s)=\frac{1000}{s^2+3s} \qquad (1)$$

Battery blocks are connected with each node (i.e. controller, sensor/actuator, interference sender and interference receiver nodes) to manage the channel transmission. Power control tasks are running in these nodes

periodically send out ping messages to other nodes to test the channel transmission. If a reply is received from the other node, then the channel is considered to be good and the transmission power will be lowered. If no reply is received from the other node, then the transmission power is increased until it saturates or a reply is received again.

The proposed WNCS model which is designed using MATLAB and TrueTime toolbox. A sub-block is implemented for each block of the WNCS model to make it easy and simple to understand as it is shown in figure (3)

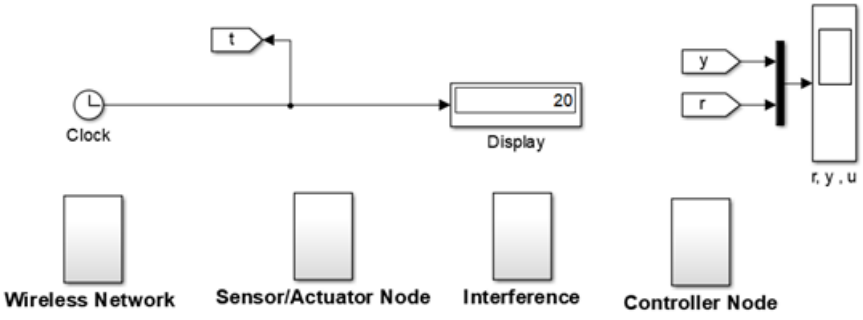

Figure (3). The Designed WNCS Model

5. Proposed PSO-PID and PSO-FOPID Controllers

The main reason to use PSO algorithm with a PID controller is to improve the step transient response of a typical stepper motor system. The obtained algorithm is called the PSO-PID controller. The PSO algorithm was invoked to find optimal values for the four controller parameters kp, ki, kd, and filter coefficient N such that the controlled system could achieve a good step response output. Each particle, P, represents the four controller parameters kp, ki, kd and N, i.e., P = [kp, ki, kd, N], where kp, ki, kd, and N are real numbers.

PSO algorithm uses a population of particles, called a swarm. Let n be the number of particles, hence, we have a swarm of size n x 4, as shown below:

$$\text{Swarm(N, 4)} = \begin{vmatrix} K_{p(1,1)} & K_{i(1,2)} & K_{d(1,3)} & N_{(1,4)} \\ K_{p(2,1)}, & K_{i(2,2)} & K_{d(2,3)} & N_{(2,4)} \\ \vdots & \vdots & \vdots & \vdots \\ K_{p(n,1)} & K_{i(n,2)} & K_{d(n,3)} & N_{(n,4)} \end{vmatrix} \qquad (2)$$

The main goal of PSO-PID algorithm is to find a set of good control parameters *kp*, *ki*, *kd*, and *N*, which can achieve a good step response. To obtain this set, the fitness function, described in (1), is used:

$$J=\int_0^\infty [w_1|e(t)| + w_4|e(t)|]dt + w_3 t_s \qquad (3)$$

Where *e(t)* is system error and *ts* is settling time of response. Using this item $(w_3 t_s)$can decrease the settling time of response, and coefficient *w3* is also important for the smooth of the response curve. Moreover, to avoid overshoot of response, the punishment of overshoot method is adopted. The punishment item $(w_4|e(t)|)$ is added to the fitness

function. Where *w4* is much larger than w1 (*w4 > w1*), the parameters choosing of *w1, w3, w4* is very important for the performance of optimization [19].

The searching procedures of the proposed PSO-PID controller is described in Algorithm (1). These steps will be repeated when the PSO method is used for FOPID controller. However, FOPID controller has five parameters to be optimized [kp, ki, kd, lamda, and mu], instead of four.

Algorithm 1 Pseudocode of the Proposed PSO-PID

1: Specify the lower and upper bounds, LB and UB respectively, of controller parameters, where: LB = [kp min, ki min, kd min, N min] and UB = [kp max, ki max, kd max, N max]. The LB and UB of controller parameters are chosen based on tuning.

2: Initialize the inertia weights, wmin and wmax, and acceleration factors, c1 and c2

3: Generate a population of n particles randomly, where each particle P consists of kp, ki, kd and filter coefficient N

4: Set the initial position of each particle, which is in the range between LB and UB, according to the following formula:

5: x(i,j)= LB (j) +rand()*(UB (j)- LB (j)), i=1 ,2 ...n, and j=1 ,2 ,3,4.

6: Set the initial velocity of each particles according to the following formula: v(i,j)=0.1 * x(i,j)

7: Let the global best particle (gbest) be the first particle x0.

8: Calculate the fitness function of each particle in the population using the fitn ess function defined in (2).

9: Determine the population best particle (pbest) which is the particle that has the minimum fitness function.

10: if fitness(pbest) ≤ fitness(gbest) then

11: gbest ← pbest

12: end if

13: while (Stop Condition is not met) do

14: Update the inertial weight

15: Update each particle Velocity according to: v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))+ c2*rand()*(gbest(1,j)- x(i,j));

16: update each particle position according to: x(i,j)=x(i,j)+v(i,j);

17: Calculate the fitness function of each particle in the population using the fitness function defined in (3).

18: Find the new pbest

19: if fitness(pbest) ≤ fitness(gbest) then

20: gbest ← pbest

21: end if

22: end while

6. Experimental Results

The sampling time of the system with three-time samplings in Interference_sender node=0.034, 0.5, 1 sec which are taken randomly for testing. The three sampling times in Interference_sender are tested for both controllers when the network is medium loaded [BWshare=0.4] and when it is highly loaded [BWshare=0.9] as shown in table 2. Three cases based on time sampling is experienced. These sampling times are 0.034, 0.09, 0.15 sec which is determined by using Shannon's sampling theorem for the transfer function in (1). From this table, the sampling time of system, sampling time in interference_sender node and BWshare, as well as the random value [w] in (4) of packet length, are also affecting the number of emulated nodes by Interference_sender node. The [w] parameter gives random values for every run for WNCS simulated model. Additionally, the number of packets and the average of Round Trip Time are measured for the packets which are sent between the controller and sensor/actuator nodes over the network. The number of emulated nodes by interference_ sender node is used to study their effect on the number of packets and average of RTT for the sending packets between the controller and sensor/actuator nodes wirelessly.

$$\text{packet\_length} = (\text{Nodes}*\text{BWshare} \times \text{maxP} \times w) \text{ byte} \quad (4)$$

In this paper, we considered the case when the time sampling of the system = 0.15 sec. The case includes a scenario when time sampling in Interference_sender node =1 sec. for all the cases, the PSO-PPID and PSO-FOPID have used the parameters listed in Table (1). Both controllers are compared based on overshoot performance criteria to examine the controller performance.

Table 1. The Proposed PSO_PID and PSO_FOPID Parameters

PARAMETERS	VALUES
Number of variables	4 for PID and 5 for FOPID
Population size	30
Acceleration factor: c1 = c2	2
Inertia weight:	wmax=0.2, wmin=0.1
The maximum number of iterations	10

6.1 SCENARIO ONE: PID AND FOPID CONTROLLERS, BWSHARE= 0.4

The number of emulated nodes by Interference_sender node is set to 14000 nodes, so the total nodes in WNCS are equal to 14003 nodes. The three nodes include the two control system nodes (Controller and Sensor/Actuator nodes) and interference receiver node. the packet_length described (4) will be ($14003 \times 0.4 \times 2304 \times w$) byte. In addition to the sensor_packet and controller_packet, there are other packets that will be sent over the network channel. While for FOPID controllers, the number of emulated nodes by Interference_sender node is set to 15000 nodes, so the total nodes in WNCS are equal to 15003 nodes. The three nodes include the two control system nodes (Controller and Sensor/Actuator nodes) and interference receiver node. The packet_length described in (4) will be ($15003 \times 0.4 \times 2304 \times w$) byte.

In addition to the sensor_packet and controller_packet, there are other packets will be sent over network channel Figure (4) shows that the system response remains stable with 14003 nodes. While for FOPID controller, the system response remains stable with 15003 nodes.

Table 2. WNCS model with the load

Sampling time of system = 0.15				
Sampling time in Interference_ Sender node	The number of nodes according to Time sampling of system, Interference_sender node, BWshare, and type of controller			
	Medium loaded network (BWshare=0.4) scenario one		High loaded network (BWshare=0.9) scenario two	
	PID	FOPID	PID	FOPID
0.034	1700	2000	640	700
0.5	7500	8100	3500	4000
1	14000	15000	7000	8000

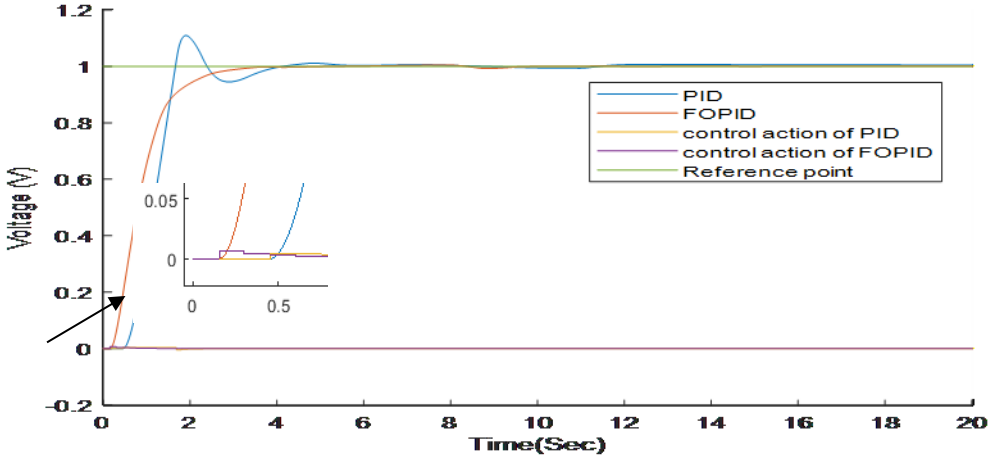

Figure (4). System Response for PID and FOPID Controllers for Ts=0.15s, BWshare=0.4 and Ts in interference_sender node=1s (Third Case, Scenario One with Load)

for 20 sec simulation time, the RTT values are shown in Figure 5, and the histogram of this RTT is shown in Figure 6, the captured packets are 134. The Two diagrams show that the RTT values are bounded between 0.0086 sec and 0.155 sec, the average of RTT=0.9562sec and overshoot=10.4207%.

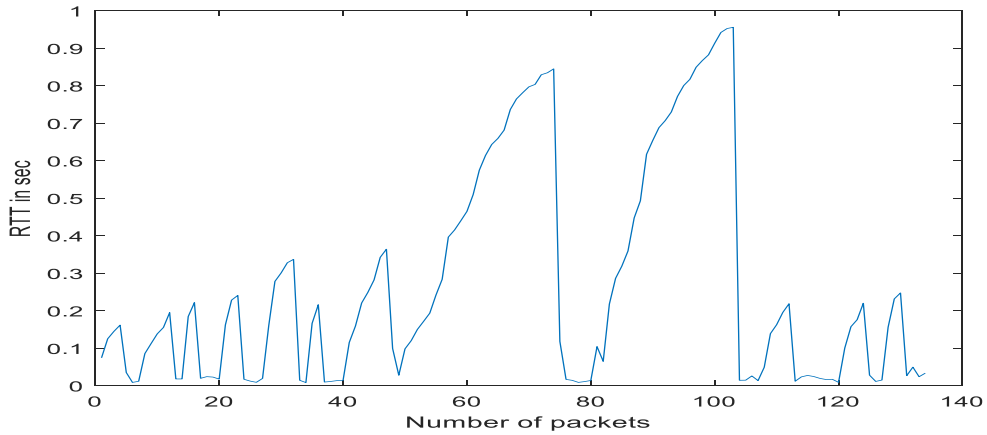

Figure (5). RTT Values for PID Controller for Ts=0.15s, BWshare=0.4 and Ts in interference_sender node=1s (Third Case, Scenario One with Load)

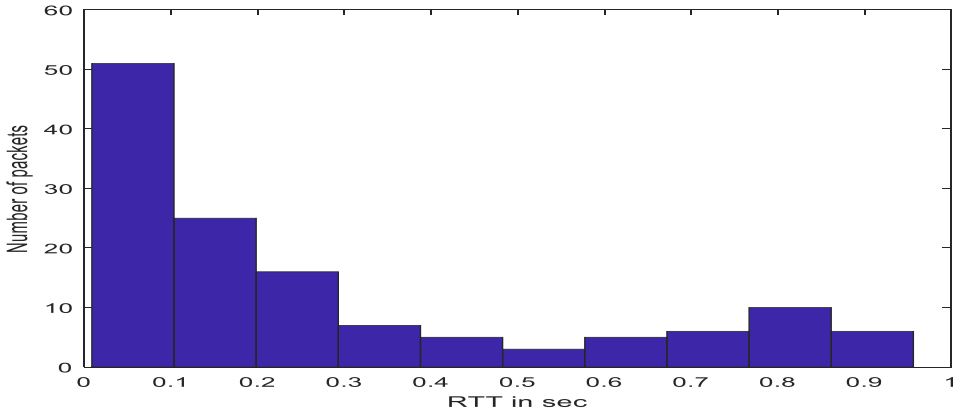

Figure (6). RTT Histogram for PID Controller for Ts=0.15s, BWshare=0.4 and Ts in interference_sender node=1s (Third Case, Scenario One with Load)

for 20 sec simulation time, the RTT values are shown in Figure 7, and the histogram of this RTT is shown in Figure 8, the captured packets are 127. The Two diagrams show that the RTT values are bounded between 0.0045 sec and 0.6604 sec, the average of RTT=0.138 sec and overshoot=0.3793%. From results above, FOPID controller is better than the PID controller because its overshoot is less than PID's overshoot.

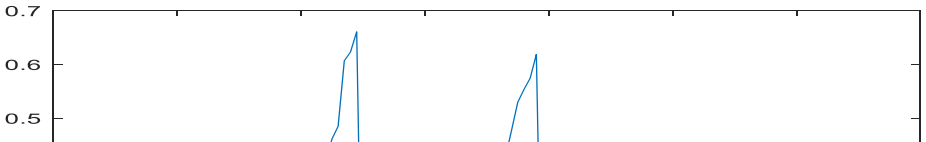

Figure (7). RTT Values for FOPID Controller for Ts=0.15s, BWshare=0.4 and Ts in interference_sender node=1s (Third Case, Scenario One with Load)

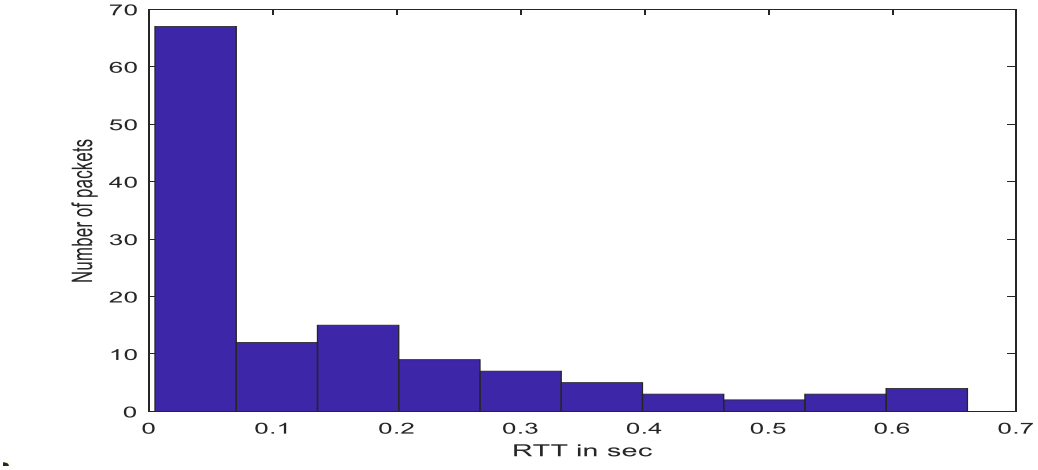

Figure (8). RTT Histogram for FOPID Controller for Ts=0.15s, BWshare=0.4 and Ts in interference_sender node=1s (Third Case, Scenario One with Load)

7. Conclusion and Future Work

In the control, communication and computing domains, WNCS attracted many researchers. The various time delays, which occurs in the measurements, represents one of the main problems in these systems as the system performance is affected and the control design process would be complicated if there is a time delay in the feedback loop. Two optimized PSO-based controllers, namely PSO-PID and PSO-FOPID, are proposed for WNCS to control the plant wirelessly.

The proposed FOPID controller is more robust than the PID controller to control the system performance to tackle the number of nodes and remain stable. The FOPID controller's highly effect is represented when time sampling of the system is 0.15 sec and time sampling in Interference_sender node is 1 sec, the controlled system by FOPID controller can handle 15000 and 8000 number of nodes when the BWshare is 0.4 and 0.9 respectively whereas the controlled system of PID controller can handle 14000 and 7000 number of nodes when the BWshare is 0.4 and 0.9 respectively.

Although the achieved results are promising, there is still room for further improvements. Using PiccSIM which is a co-simulator for network and control system simulation in a networked control system (NCS), is one possible enhancement. It is intended for research on NCS or Wireless NCS (WNCS). This co-simulator is used to integrate the NS-2 as a network simulator and MATLAB. Moreover,

Implementing different types of wireless protocols such as Wi-Fi (802.11e, 802.11ac, 802.11n, etc) instead of 802.11b to the same WNCS model and see its effects on this model in terms of the number of nodes for both controllers.

References

[1] Willig, A. et al. (2005). Wireless technology in industrial networks. *Proceedings of the IEEE*. *93*, 1130-1151.

[2] Zhang, W. et al. (2001). Stability of networked control systems. *IEEE control systems magazine*.*21*, 84-99.

[3] Xue, Y. et al. (2006). Self-tuning of PID parameters based on the modified particle swarm optimization. *2006 4th IEEE International Conference on Industrial Informatics*.870-873.

[4] Deepyaman, M. et al. (2008). Tuning PID and PIλ Dμ controllers using the integral time absolute error criteria.*4th International Conference on Information and Automation for Sustainability ICIAFS*. 457-462.

[5] Meng, L., & Xue, D. (2009). Design of an optimal fractional-order PID controller using multi-objective GA optimization. *2009 Chinese Control and Decision Conference*. 3849-3853.

[6] Cao, J. Y. et al. (2005). Optimization of fractional order PID controllers based on genetic algorithms. *2005 international conference on machine learning and cybernetics*. 5686-5689.

[7] Petráš, I., Dorčák, L., & Koštial, I. (1998). Control quality enhancement by fractional order controllers. *Acta Montanistica Slovaca*. 1998, 143-148.

[8] Zamani, M. et al. (2007). FOPID controller design for robust performance using particle swarm optimization. *Fractional Calculus and Applied Analysis*. *10*, 169-187.

[9] Mohamed, M. J., & Khashan, M. A. (2014). Comparison Between PID and FOPID Controllers Based on Particle Swarm Optimization. The Second Engineering Conference of Control, Computers and Mechatronics Engineering ECCCM2.

[10] Alwahab, D. (2016). Enhancement of WNCS response using packet priority. *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications AIC-MITCSA.*1-6.

[11] Isra‟a, L., & Osama, A. (2015). Gain Scheduling PID Controller For Networked Control System with Random Delay. *International Journal of Enhanced Research in Science Technology & Engineering.* 4, 184-193.

[12] Mais, M. & Osama, A. (2017). Evaluating a ZigBee Network with SMC for Hard and Concurrent Parameter Variations. *Journal of Information Engineering and Applications.* 7.

[13] Vipul, R. & Dhwanit, C. (2016). Simulation of Wireless Network Using TrueTime Toolbox. *International Research Journal of Engineering and Technology.* 3.

[14] Tarmizi, W. et al. (2016). A Particle Swarm Optimization-PID controller of a DC Servomotor for Multi-Fingered Robot Hand. *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation ROMA.*1-6.

[15] Cervin, A. et al. (2010). TrueTime 2.0 beta-Reference manual. *Department of Automatic Control, Lund University*. Boughanmi, N. et al. (2008). Wireless networked control system using IEEE 802.15. 4 with GTS. *2nd Junior Researcher Workshop on Real-Time Computing, JRWRTC 2008.* 21-25.

[16] Millan, Y. et al. (2011). A wireless networked control systems review. *IX Latin American Robotics Symposium and IEEE Colombian Conference on Automatic Control, 2011 IEEE*. 1-6.

[17] Cuenca, Á. Et al. (2011). A delay-dependent dual-rate PID controller over an ethernet network. *IEEE Transactions on Industrial Informatics*. *7*, 18-29.

[18] Li, X. et al. (2007). PSO algorithm based online self-tuning of PID controller. *2007 International Conference on Computational Intelligence and Security CIS 2007.* 128-132.